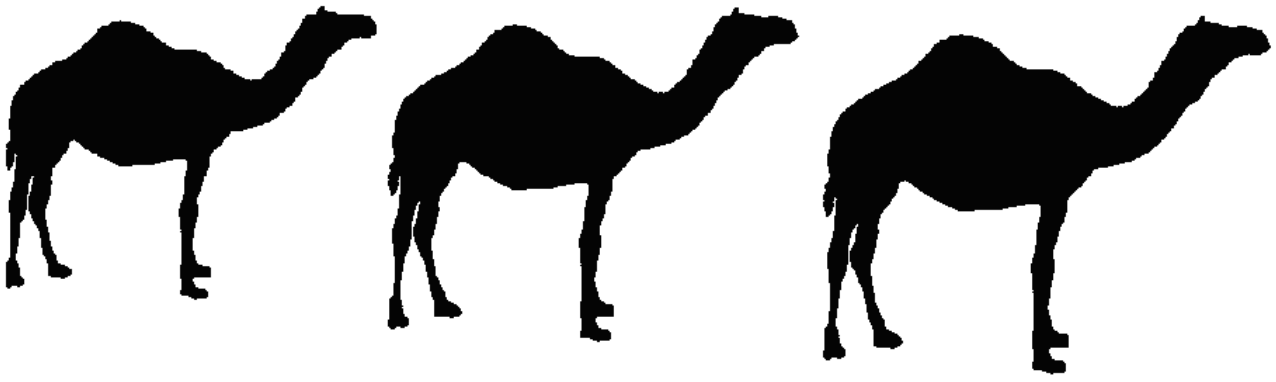
[K]AML



Array Manipulation Language
Final Report

***K**aori Fukuoka* *kf2284@columbia.edu*
***A**nkush Goel* *ag2812@columbia.edu*
***M**aninder Singh* *ms3770@columbia.edu*
*Mayur **L**odha* *mdl2130@columbia.edu*

COMS 4115: Programming Languages and Translators, Fall 2008: Prof. Stephen Edwards.

TABLE OF CONTENTS

1. Introduction

1.1 Motivation

Array is one of the fundamental data structures in many programming languages. It allows one to store many values under the same name. Thus, it can be defined as an ordered, integer indexed collection of objects. In computer science, array has a wide range of applications starting with implementation of mathematical vectors, matrices to various data structures like heap, hash table, stacks, queues, etc. It also plays a major role in many problem solving algorithms. Therefore, array manipulations form an important as well as error prone part of these algorithms.

So far, no language has direct support for array manipulations as they have for integer or other primitive data type. Some array programming languages like APL do provide powerful set of inbuilt functions, but it makes programming difficult with its distinctive mathematical form and character set. For functionalities which have support for arrays, it thus becomes important and necessary to have a language which is handy and allows one to write more effective and succinct code.

1.2 Description

[K]AML is an expressive and concise Array Manipulation language which features rich set of operations on Arrays. Unlike other structured programming languages, [K]AML treats array as a primitive data type. Thus, it would not require a beginner to think of arrays as a collection of data but as a single data-type in itself.

[K]AML has a high-level syntax and semantics supporting features like concatenation, copying, reverse, split, etc. Arrays and variables are initialized at first point of use and have global life and visibility from the point where they have been defined. All numerals in [K]AML program are treated as integers. Array manipulations are done using high level operators. This makes it possible to express a computable function using just an expression in a single line of code, reducing the potential number of loops and allowing to writing concise and compact programs. Familiar and intuitive control flow structure provides functionality for iteration and selection to control flow of program execution. The language provides error handling for syntactical errors but does not support exception handling mechanism. User defined functions enable efficiency of code.

In short, [K]AML follows a consistent structure making language simple to program and understand. It is designed to be clear and intuitive to anyone familiar with structured programming.

1.3 Language Features

Data Type

[K]AML does not require explicitly declaring a type for a variable. A variable if declared with square brackets [ ] is explicitly considered as an array. Else it would be implicitly treated as an integer variable.

Unlimited size Arrays

[K]AML supports one dimensional and two dimensional arrays. The index of first element of an array is 0. Declaration of a single dimension array does not require one to specify the size of the array. The size of the array is implicitly computed from the array initialization list specified as part of array declaration. For two dimensional arrays, size for the second dimension has to be specified. The size for the first dimension is implicitly computed from the array initialization list specified as part of array declaration. Since arrays can be declared at any point in program, empty arrays cannot be declared.

Padding elements of an Array

Implicit padding of array elements is done incase of two dimensional arrays. If the number of elements in the inner array is not equal to the specified length (indicated in the second dimension of the array), then the inner array is implicitly padded with 0 to make length equal to the specified length.

Dynamic length Arrays

Array once declared, its length can be increased or decreased by performing specific operations. If a new element is added to an existing array, its length is increased by one. Similarly, length of an array decreases by one on deleting an element from an array.

Selection on Demand

[K]AML allows operations to be performed on a specific element or group of elements. It allows one to specify the range or list of array elements to be accessed from an array. The array dimension can include list or range of array indexes, thus allowing specific operation to be performed on selected indexed elements. At most one range selection can appear in the first dimension and any number of individual elements can be selected. These indexes and range are specified as comma separated list and can appear only in the first dimension of an array. The selection list if specified, should index more than one element of an array.

Operators

[K]AML supports basic operators to perform conditional and arithmetic operations. These operators are applicable to both variables and constants. In addition, various operators are defined on arrays to support array manipulations. The complete listing of operators is given in the language reference manual.

Control Flow

To control flow of program, [K]AML supports for loop for iteration and if/else loop for branching.

User-defined Functions

Code reusability can be achieved by user defined functions. [K]AML also supports recursions, a key for various divide and conquer algorithms implementation.

Output

[K]AML has a built in function *show* which provides access to the standard output.

1.4 Sample Programs

1.4.1. Implementation of Inverse of a Matrix

```
function display([][]c)
{
   for(i=0;i<#[]c;i++)
   {
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);
        }
   }
}
```

```
function power(y)
{

          z=1;
          for(x=1;y>0;y--)
          {
                    z=-z;
          }
          return z;
}

function deter2([][]r)
{
    temp=0;
    temp=[0][0]r*[1][1]r-[0][1]r*[1][0]r;
    return temp;
}

function determinant([][]b,m)
{
 sum=0;
 [][3]c={{0,0,0},{0,0,0},{0,0,0}};    //Its size should be atleast (n-1)*(n-1)
 ?(m==2)
  {
          sum = fun deter2([][2]b);
  }
 !
  {
          for(p=0;p<m;p++)
          {
                     h=0;
                     k=0;
                     for(i=1;i<m;i++)
                     {
                              for(j=0;j<m;j++)
                              {
                                        ?(j!=p)

                                        {
                                                  [h][k]c=[i][j]b;
                                                  k=k+1;
                                                  ?(k==m-1)
                                                  {
                                                            h=h+1;
                                                            k=0;
```

```
                                                }
                                        }
                                }
                        }
                        temp=fun determinant([][]c,m-1);
                        t=fun power(p);
                        sum=sum+[0][p]b * t * temp;
                }
        }
return sum;
}

function cofactor([][]a,i,j)
{
        []x={0,0};
        [][2]y={{0},{0}};
        z=i+1;
        [][]a->z;
        l=0;
        for(k=0;k<(#[][]a/#[]a);k++)
        {
                ?(k!=j)
                {
                        []x=[][k]a;
                        [][l]y=[]x;
                        l=l+1;
                }
        }
        return [][]y;
}

 [][3]a={{1,1,2},{1,1,5},{1,2,1}};
 [][3]f={{0},{0},{0}};
 for(i=0;i<#[]a;i++)
  {
        for(j=0;j<(#[][]a/#[]a);j++)
        {
                t1=0;
                [][]t=fun cofactor([][]a,i,j);
                t1=fun power(i+j);
                 for(m=0;m<#[]t;m++)
                {
                        for(n=0;n<(#[][]t/#[]t);n++)
                        {
                                [m][n]t=([m][n]t*t1);
                        }
                        }
                        t5=fun deter2([][]t);
                        [i][j]f=t5;
            }
 }
 x=fun determinant([][]a,3);
 for(m=0;m<#[]f;m++)
            {
                        for(n=0;n<(#[][]f/#[]f);n++)
                        {
                                    [m][n]f=([m][n]f/x);
                        }
            }
 show("The input matrix is ");
 fun display([][]a);
 show("\n\nThe inverse of the matrix is ");
 fun display([][]f);
```

The corresponding output is:

```

The input matrix is
1 1 2
1 1 5
1 2 1

The inverse of the matrix is
3 1 0
1 0 0
-1 -1 0
```

1.4.2. Implementation of Stacks and Queues

```
function display([]x)
{
 for(i=0;i<#[]x;i++)
 {
            show([i]x);
 }
  show("\n");
}

function enqueue([]a, c)
```

```
{
[]a<-{c};
show("The status of queue after Enqueue ", c,"\n");
fun display([]a);
return []a;
}

function dequeue([]a)
{
[]a->1;
show("The status of queue after Dequeue \n");
fun display([]a);
return []a;
}

function push([]b, c)
{
[]b<-{c};
show("The status of stack after Push ",c,"\n");
fun display([]b);
return []b;
}

function pop([]b)
{
x=[#[]b-1]b;
[]b->#[]b;
show("The status of stack after Pop ","\n");
fun display([]b);
show("The popped element is ",x,"\n");
show("\n");
return []b;
}

[]a={1,2,3,4,5};
[]b={1,2,3,4,5};
show("Initial queue status:\n");
fun display([]a);
[]a=fun enqueue([]a,4);
[]a=fun enqueue([]a,4);
[]a=fun enqueue([]a,4);
[]a=fun enqueue([]a,4);
[]a=fun dequeue([]a);
[]a=fun dequeue([]a);
[]a=fun dequeue([]a);
 show("\n"); show("\n");
show("Initial stack status:\n");
fun display([]b);
[]b=fun push([]b,4);
[]b=fun push([]b,4);
[]b=fun pop([]b);
[]b=fun pop([]b);
[]b=fun pop([]b);
```

The corresponding output is:

```
Initial queue status:
1 2 3 4 5
The status of queue after Enqueue 4
1 2 3 4 5 4
The status of queue after Enqueue 4
1 2 3 4 5 4 4
The status of queue after Enqueue 4
1 2 3 4 5 4 4 4
The status of queue after Enqueue 4
1 2 3 4 5 4 4 4 4
The status of queue after Dequeue
2 3 4 5 4 4 4 4
The status of queue after Dequeue
3 4 5 4 4 4 4
The status of queue after Dequeue
4 5 4 4 4 4


Initial stack status:
1 2 3 4 5
The status of stack after Push 4
1 2 3 4 5 4
The status of stack after Push 4
1 2 3 4 5 4 4
The status of stack after Pop
1 2 3 4 5 4
The popped element is 4

The status of stack after Pop
1 2 3 4 5
The popped element is 4

The status of stack after Pop
1 2 3 4
```

```
The popped element is 5
```

1.4.3. Implementation of Transpose of a Matrix

```
function show2darray([][]x)
{
show("\n");
for(i=0;i<#[]x;i++)
{
 for(j=0;j<(#[][]x/#[]x);j++)
 {
  show([i][j]x, " , ");
 }
show("\n");
}
}

function transpose([][]x,[][]y)
{
i=0;
for(i=0;i<#[]x;i++)
{
 []temp = [i][]x;
 [][i]y = []temp;
}
return [][]y;
}

[][4]x ={{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
[][4]y ={{0},{0},{0},{0}};

show("Original array is  [4][4]x","\n");
fun show2darray([][]x);

[][]x = fun transpose([][]x,[][]y);

show("After Transpose array is : [4][4]x","\n");
fun show2darray([][]x);
```

The corresponding output is:

```
Original array is  [4][4]x

1  , 2  , 3  , 4  ,
5  , 6  , 7  , 8  ,
9  , 10  , 11  , 12  ,
13  , 14  , 15  , 16  ,

After Transpose array is : [4][4]x

1  , 5  , 9  , 13  ,
2  , 6  , 10  , 14  ,
3  , 7  , 11  , 15  ,
4  , 8  , 12  , 16  ,
```

1.4.4 Implementation of Determinant of Matrix

```
[][4]a={{1,2,1,2},{1,1,1,5},{1,2,1,6},{9,1,2,0}};     //Initilize the n*n matrix

function display([][]c)
{
 for(i=0;i<#[]c;i++)
  {
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);
        }
  }
}

function power(y)
{
        z=1;
        for(x=1;y>0;y--)
        {
                z=-z;
        }
        return z;
}

function deter2([][]r)
{
temp=[0][0]r*[1][1]r-[0][1]r*[1][0]r;
return temp;
}

function determinant([][]b,m)
{
 sum=0;
 [][3]c={{0,0,0},{0,0,0},{0,0,0}};     //Its size should be atleast (n-1)*(n-1)
 ?(m==2)
  {
          sum = fun deter2([][2]b);
  }
 !
  {
          for(p=0;p<m;p++)
          {
                    h=0;
                    k=0;
                    for(i=1;i<m;i++)
                    {
                           for(j=0;j<m;j++)
                           {
                                   ?(j!=p)
                                   {
                                           [h][k]c=[i][j]b;
                                           k=k+1;
                                           ?(k==m-1)
                                           {
                                                   h=h+1;
                                                   k=0;
                                           }
                                   }
                           }
                    }
                    temp=fun determinant([][]c,m-1);
                    t=fun power(p);
                    sum=sum+[0][p]b * t * temp;
           }
  }
return sum;
}

show("The input matrix is \n");
fun display([][]a);
x=fun determinant([][4]a,4);    //Call the function to calculate the determinant
show("\nThe determinant is ",x);
```

The corresponding output is:

```
The input matrix is

1 2 1 2
1 1 1 5
1 2 1 6
9 1 2 0

The determinant is -28
```

2. Language Tutorial

[K]AML program mainly consists of a sequence of statements.

2.1 Show Statement

A basic "Hello World" program can be written in [K]AML in a single statement as:

show("Hello World !!!");

This prints "Hello World !!!" to the standard output.

show is an executable statement which prints the input string to the standard output. In addition, it can also be used to print data of basic types.

show("Sum of 2 and 3 is : ", 2+3);

This prints " Sum of 2 and 3 is : 5".

show("My favorite number is : ",7(3+3));*

This prints " my favorite number is : 42".

2.2 Array Declaration and Padding

[K]AML supports single and two dimensional arrays.

[ ]a = {1,2,3,4,5,6,7,8,9,10};

[ ][2]b = { {1,2}, {3,4}, {5}, {6} };

The first statement declares and initializes a single dimensional array *a* whose size is implicitly computed as 10. The second statement declares and initializes a two dimensional array *b* whose size for second dimension is 2. The third and the forth inner array of array *b* have just one element. Thus both these inner arrays are padded with one 0 to make them equal to size 2. Thus internally, array *b* will be recognized as:

*[ ][2]b = { {1,2}, {3,4}, {5,**0**}, {6,**0**} }*

2.3 Array Selection

[K]AML allows one to access selected elements of an array.

[ ]a = {1,2,3,4,5,6,7,8,9,10};

[ ]b = [0,3..6,8,9]a;

The first statement declares and initializes an array *a* whose size is implicitly computed as 10. The second statement declares an array *b* and is initialized with the elements at position 0, 3, 4, 5, 6, 8, 9 of array *a*, thus computing its size or array *b* to be 7.

```
[ ]b = {1,4,5,6,7,9,10};
```

2.4 For Loop

[K]AML allows *for* statement to support iteration.

```
for (i = 0; i<#[ ]a; i++)
{
 show( [i ]b , " :: ");
 }
```

This statements prints all the elements of the array *b*. # is an operator which specifies the size of the array. Here *#[ ]a* has value 7. Thus, the above *for* loop prints:

```
1 :: 4 :: 5 :: 6 :: 7 :: 9 :: 10
```

2.5 Conditional Statement

[K]AML allows ? ! statement to control flow of program execution.

```
? (a > b)
{
        show(" a is greater than b");
}
!
{
        show("b is greater than a");
}
```

This code prints *" a is greater than b"* if the condition *a > b* evaluates to true else it prints *" b is greater than a"*.

2.6 Set Operations

[K]AML supports basic Set Operations on arrays. These are binary operations which includes Union, Intersection and Difference of arrays. Set operations can only be applied to compatible arrays. Two arrays are said to be compatible if both have same number of dimensions and for two dimensional arrays, both have same length for the second dimension.

[ ][2]a = { {1,2}, {3,4}, {5,6} };

[ ][2]b = { {5,6}, {7,8}, {9}, {10} };

Here array *a* and *b* are both two dimensional arrays with second dimension of length 2. Thus, they are compatible for Set operations.

The Union operation on arrays is specified as:

[ ][2]c = %+([ ][ ]a, [ ][ ]b);

This statement declares and initializes array *c* with Union of array *a* and *b*. Thus array *c* now becomes:

[ ][2]c = { {1,2}, {3,4}, {5,6}, {7,8}, {9,0}, {10,0} };

The Intersection operation on arrays is specified as:

[ ][2]c = %=([ ][ ]a, [ ][ ]b);

This statement declares and initializes array *c* with Intersection of array *a* and *b*. Thus array *c* now becomes:

[ ][2]c = { {5,6} };

The Difference operation on arrays is specified as:

[ ][2]c = %-([ ][ ]a, [ ][ ]b);

This statement declares and initializes array *c* with Difference of array *a* and *b*. Thus array *c* now becomes:

[ ][2]c = { {1,2}, {3,4} };

Array selection list is applicable to set operations as well. Note that selection can only be applied on the first dimension of the array.

[ ][2]c = %=([0..1][ ]a, [ ][ ]b);

Here array *a* would be considered as:

[ ][2]a = { {1,2}, {3,4} };

2.7 Reverse Array

[K]AML supports reversing the elements of an array. The source array remains the same. The reversed array is assigned to the destination array. Both the source and the destination arrays need to be compatible.

[ ][2]c = - [ ] [ ]b;

This makes array *c* to be:

[ ][2]c = { {10,0}, {9,0}, {7,8}, {5,6} };

2.8 Copy Array

A duplicate copy of an array can be made by simply assigning one array to another. Both the source array and destination array needs to be compatible.

[ ][2]a = { {1,2}, {3,4}, {5,6} };

[ ][ ]b = [ ][ ]a;

Here, array *b* is declared with elements same as that of array *a.* Part of array can be selected to initialize it to destination array.

[ ][2]b = [0,1][ ]a;

This makes array *b* to be:

[ ][2]b = { {1,2}, {3,4} };

2.9 Size of Array

Size of an array can easily be computed using the # operator.

[ ][2]a = { {1,2}, {3,4}, {5,6} };

#*[ ][ ]a* returns 6 as total number of elements in the array.

#*[ ]a* returns 3 as length of the first dimension of array.

2.10 Sum Array

[K]AML supports computing the sum of elements of an array.

[ ][2]a = { {1,2}, {3,4}, {5,6} };

+*[ ][ ]a* returns 21 as sum of all the elements of an array.

+*[0][ ]a* returns 3 as sum of the elements of the first inner array.

+*[ ][0]a* returns 12 as sum of all the first elements of all inner arrays.

2.11 Concatenate Arrays

Array concatenation is a binary source operation and can be done in a single step.

[ ][2]a = { {1,2}, {3,4}, {5,6} };

[ ][2]b = { {5,6}, {7,8}, {9}, {10} };

[ ][ ]a ++ [ ][ ]b;

Array *b* remains unchanged. Array *a* now becomes:

[ ][2]a = { {1,2}, {3,4}, {5,6}, {5,6}, {7,8}, {9,0}, {10,0} };

2.12 Array Access

Individual elements of an array can be easily accessed and updated by specifying the array index.

[ ][2]a = { {1,2}, {3,4}, {5,6} };

[0][0]a returns 1 and *[0][1]a* returns 2.

2.13 Insert elements into Array

Array once declared, it can be made to grow dynamically by inserting additional elements into an array.

[ ]a = {1,2,3,4,5};

[ ]a <- {6,7,8};

The second statement inserts elements 6,7 and 8 into array *a.* Thus, array *a* now becomes:

[ ]a = {1,2,3,4,5,6,7,8};

For two dimensional arrays:

[ ][2]b = { {1,2}, {3,4}, {5,6} };

[ ][ ]b <- { 7,8,9 };

This inserts {7,8} and {9,0} into array *b*. Thus array *b* now becomes:

[ ][2]b = { {1,2}, {3,4}, {5,6}, {7,8}, {9,0} }

2.14 Delete elements from Array

Array once declared, its elements can be deleted by specifying the *nth* element to be deleted. One element can be deleted at a time and the length of the array is decreased by one on each deletion.

[ ]a = {11,12,13,14,15};

[ ]a -> 2;

This deletes the second element from the array, thus deleting *12* from array *a*.

[ ][2]b = { {1,2}, {3,4}, {5,6} };

[ ][ ]b -> 1;

This deletes the first element of the array, thus deleting *{1,2}* from array *b*.

2.15 Defining Functions

[K]AML allows user defined functions. Functions may be recursive and may return value.

```
function add(a,b)
{
        return a+b;
}
```

This defines a function *add* with two parameters *a* and *b*. It returns the sum of *a* and *b*.

3. Language Reference Manual

3.1Lexical Conventions

[K]AML is a free-form language which includes identifiers, keywords, constants, separators and operators as tokens. Spaces, tabs and newlines only serve to separate tokens of the language and are ignored during the compilation.

3.1.1 Comments

[K]AML supports both single line and multi line comments. Single line comments begin with two forward-slash characters ”//” and terminate at the end of the line in which they appear. Multi-line comments begin with the characters ”/*” and terminate after the first instance of the characters ”*/” is encountered.

3.1.2 Identifiers

Identifier is a non-empty string consisting of a sequence of letters and numbers. Letters include lowercase and uppercase alphabets from the ASCII set. Identifiers are case sensitive with the first character always being a letter.

3.1.3 Keywords

The following identifiers are the reserved keywords and cannot be used otherwise:

show **for** **function** **fun** **return**

3.1.4 Constants

[K]AML supports two types of constants:

3.1.4.1 Integer Constants

An Integer constant consists of an optional plus “+'” sign or minus “-“ sign followed by a string of decimal digits.

3.1.4.2 String Constants

String constants consist of a sequence of characters enclosed in double quotes. They are only used within *show* statement. Everything within the quotes will be printed to the standard output. Quotes are not considered as part of the string. A double-quote character may be included in the string by preceding quote with "\". [K]AML recognizes the following escape sequences:

n	as newline
t	as horizontal tab

3.1.5 Separators

A single semicolon character **";"** is used as a separator. It indicates end of an executable statement.

3.2 Types

Every identifier in [K]AML has a type which determines the meaning of the value stored in the identifier's storage. [K]AML supports two types:

3.2.1Integers

The basic type in [K]AML is an integer which consists of one or more digits in sequence.

3.2.2Arrays

An array contains a sequence of elements of the same type. In [K]AML, an array can contain only basic type. The first element of an array is indexed at 0. Individual elements of an array can be addressed by specifying the index of the element.

3.3 L-values

L-value is an expression which refers to an object to which a value can be assigned. Any identifiers of basic type or array name can be a valid modifiable L-value.

3.4 Expressions

In [K]AML, expressions are executed in a sequential order. They are combination of various unary and binary operations performed on identifiers, constants and arrays. Binary operations include basic mathematical, conditional and comparison operations. In addition, array manipulation operations like Set operations, Array Concatenation are some of the binary operations on arrays. Unary operations on arrays include Array Reverse, Sum Array, Insert, Delete, etc. Function calls also forms a valid expression. All operators except assignment can accept expressions as their arguments. Assignment can only accept expressions as a right hand side argument. Following points illustrates in detail.

3.4.1 Unary Operations

The basic unary operations that can be performed on an expression includes "-", "++" and "--". These operators perform sign negation, increment by one and decrement by one respectively. Examples of expressions including these unary operators are:

a = - 2;
a ++;
a--;

Other unary operations on arrays includes "-", "+", "#", "<-" and "->". These operators perform array reverse, array sum, length of array, array insert and array delete respectively. Examples of array expressions including these unary operators are:

- [ ]a;
+[ ]a;
#[ ]a;
[ ]a <- {1,2,3};
[ ]a -> 2;

3.4.2 Binary Operations

The basic mathematical binary operations that can be performed on an expression includes "+", "-", "*" and "/". These operators perform mathematical addition, subtraction, multiplication and division respectively. Examples of expressions including these operators are:

*a = 1 + 2 * 3 – 5;*
*a = 2 * ((10 / 5) + 4);*

The supported binary comparison and logical operators are "<", ">", "<=", ">=", "= =", "!=", "&&" and "||". These operators mean less-than, greater-than, less-than-equal-to, greater-than-equal-to, equal-to-equal-to, not-equal-to, boolean-and and boolean-or respectively. Examples of expressions including these operators are:

a > b
a <= b
a >= b && c != 0
a < b || c = = 1

Other binary operations on arrays includes “++”, “%+”, “%=” and “%-“. These operators perform concatenation, set union, set intersection and set difference on arrays respectively. Examples of array expressions including these operators are:

```
[ ][ ]a ++ [ ][ ]b;
% + ([ ][ ]a, [ ][ ]b);
% = ([ ][ ]a, [ ][ ]b);
% - ([ ][ ]a, [ ][ ]b);
```

3.4.3 Function Calls

Function call allows one to execute a subroutine by specifying arbitrary number of function parameters. [K]AML requires one to specify keyword *fun* before the function call. The arguments of a function call may have a name associated with it to match it up with the corresponding parameter or it can simply be constants. [K]AML also allows a function to return a value. Both, parameters and return value could be basic type, constants or arrays. It is also valid in [K]AML not to make an assignment to a function call that returns a value. Examples of function calls are:

```
fun printMessage( );
fun push(c);
a = fun pop( );
b = fun add(1,2,a,b,c);
[ ][ ]c = fun multiply([ ][ ]a, [ ][ ]b);
```

3.5 Statements

Statements in [K]AML are sequentially executed in the order in which they have been written in the program. It is simply an expression terminated by a semicolon.

3.5.1 Conditional Statements

Conditional statements are composed of a “?” statement optionally followed by “!” statement. “?” is an operator used for conditional execution. The block of statements associated with the “?” are executed once if the condition evaluates to be true. If the condition specified within the “?” statement evaluates to be false, then the corresponding “!” block of statements is executed if one exist. The “!” block may further contain a “?” statement, thus providing nested conditional blocks. Example of conditional statement is:

```
? (a>b)
{
        show(“ a is greater than b”);
}
```

```
!
{
	show("b is greater than a");
}
```

3.5.2 Function Definition

A function definition consists of the *function* keyword followed by an identifier and an argument list. The identifier specifies the function name. The argument list could include comma separated list of basic type, constants and arrays. The function body consists of a left curly bracket followed by zero or more statements followed by a right curly bracket. Functions may or may not have a *return* statement. Language also supports recursive functions. Examples of functions are:

```
function printMessage()
{
	show("This is a function");
}

function add(a,b)
{
	return a+b;
}

function addFirstElementOfArray([ ]a, [ ]b)
{
	return [0]a + [0]b;
}
```

3.5.3 Return Statement

The return statement consists of a *return* keyword followed by an expression. It is an optional statement that can only appear once inside a function definition and can only be the last statement of the block. Once a return statement is encountered, execution of the function halts and the return value of the expression following the return keyword is made available to the context from which the function call was made. Examples of return statements are:

```
return 0;
return 2+a;
return [ ][ ]a;
```

3.5.4 Loop Statement

The *for* statement is a loop statement used to execute statements within the block several times depending upon a condition. It has the following syntax:

```
for (expression 1; expression 2; expression 3)
{

        zero or more statements
}
```

The expression 1 in the loop is used for initialization, expression 2 specifies a condition which is evaluated prior to each iteration and expression 3 specifies an increment or decrement action that is performed after each iteration and before the evaluation of expression 2. The execution of the statements within the *for* block continues for each iteration until expression 2 evaluates to false. Example of for loop is:

```
for (i=0; i<10; i++)
{
        show(i);
}
```

4. Project Plan

4.1 Processes

Once the group was formed, we had weekly meetings to discuss various project related issues. We planned and evenly distributed the tasks throughout that were needed to complete the project. The initial planning and language specification of the project was mostly done at the team meeting with brainstorming ideas to decide the basic operations of our language. This allowed each one to share his/her ideas and to discuss on the fundamental operations. The initial code for lexer and parser was developed at the team meeting for a small part of the language. This allowed everyone to understand the flow and coding style for O'caml. After we successfully implemented a program for basic arithmetic operations, tasks were assigned to each team member to implement the part of the language. Once all the basic operations were implemented, we started working on arrays. To start with, we made the language work for single dimension arrays. This was later enhanced for two dimensional arrays. While we still met every week to discuss problems and take updates, we also communicated through email frequently to inform any new updates in code, any issues found, or to setup time for the next meeting. Fundamental tasks were distributed with higher priority and given rough deadlines in short periods of time so that we could start working on enhancing the code for the following week. We followed an iterative process while implementing the Interpreter. Testing was also being done along with coding so that errors can be caught at an early stage and fixed. Once sufficient implementation was done, one of us focused on testing each operation to check any loop holes in the language and inform the respective team member who implemented that part of the code so that bugs can be resolved.

4.2 Programming Style Guide

The following is a list of programming style we used in our project.

- Give relevant comments to make the code understandable to all the team members.
- Regularly taking updates of the code to be in sink with the team.
- Ensuring that all the conflicts are resolved while merging or committing the files.
- Following a standard extension .kaml for all the input files.
- Used clear and relevant variable and function names throughout the code.
- Used C language-like notation wherever possible to give a clear understanding.
- Informing the team whenever any changes are made to files in the repository.
- Giving relevant comments while committing any file in the repository.
- Inform the respective member if any bug is found in his/her part of code.
- Writing relevant test cases to support exception handling for all invalid patterns.

4.3 Project Timeline

Date	Tasks
09/10/2008	Team Formation, Brainstorm
09/21/2008	Brief language syntax, Finalize proposal
09/24/2008	Project Proposal submitted
09/25/2008	Project Feedback
09/29/2008	Started working on LRM
10/10/2008	Lexer and Parser, Tasks distributed for LRM and grammar
10/12/2008	Basic Grammar completed
10/14/2008	Grammar finalized, LRM draft
10/19/2008	Finalize LRM
10/22/2008	LRM Submitted
10/23/2008	LRM feedback
10/29/2008	Midterm Exam
11/01/2008	Grammar revision, AST and Interpreter implementation
11/06/2008	Worked on basic operations in Interpreter, SVN running
11/10/2008	Lexer and Parser completed
11/16/2008	Merging and more coding in Interpreter, Symbol tables
11/21/2008	Basic array operations working
11/24/2008	Enhancing array-related operations, Exception handling
12/02/2008	Debugging
12/08/2008	Final Exam
12/12/2008	Testing
12/17/2008	Work on final report, test cases and sample programs
12/19/2008	Final Presentation

4.4 Roles and Responsibilities

The following table shows the roles and tasks that were assigned to each team member. While every member participated in almost every aspect of the projects, these are the parts that each member had more concentration to work on.

Name	Responsibilities
Kaori Fukuoka	Lexer, Interpreter, Exception Handling, Report
Ankush Goel	Interpreter, Exception Handling, Test cases, Algorithms
Maninder Singh	Parser, Interpreter, Test cases, Algorithms
Mayur Lodha	Parser, Interpreter, Optimization, Testing, Report

4.5 Development Environments

The entire project is implemented in functional programming language namely O'caml. The Lexer, Parser, Tree walker and Interpreter are all written in O'caml.

We used Cygwin v1.5.25 (http://www.cygwin.com/) and OcamlWinPlus v1.9RC4 (http://caml.inria.fr/index.en.html) for compiling and testing the code.

All of the source code and project log were maintained utilizing the Subversion system provided by Google Code (http://code.google.com/) and the Client QSvn (http://www.anrichter.net/projects/qsvn).

4.6 Project Log

We recorded project log automatically with the Subversion service. Please refer to the part 8, "Appendix – 8.2 Project Log" for the complete log of the project.

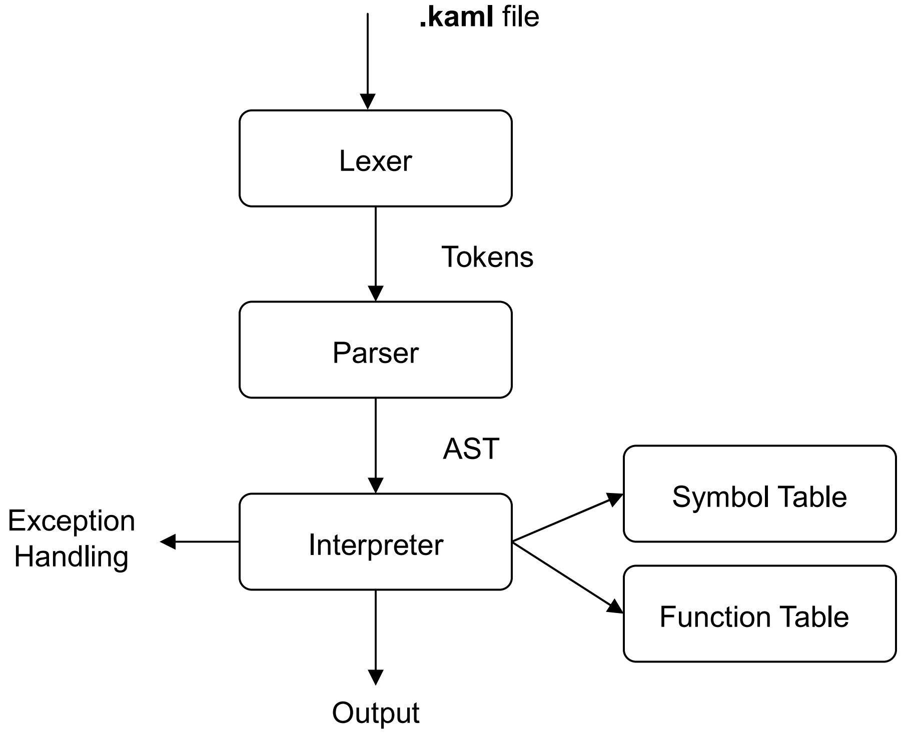
5. Architectural Design

[K]AML is an Array Manipulation Language. The interpreter for [K]AML can be viewed in two phases:

Phase 1 consists of the lexer, parser and the AST walker.
Phase 2 consists of the implementation for data types, expression evaluation and type checking.

5.1 Block diagram

The following architectural diagram shows the high level interpreter of [K]AML.



5.2 Description of Architecture

The major components of [K]AML are the lexer, the parser and the interpreter. The flow of information between these components is shown in the above block diagram.

The lexer takes in the **.kaml** file as input and perform lexical analysis of the input file. The lexical analyzer reads this input character stream and produces a stream of tokens. The parser receives the tokens and analyses the structure of the program. It checks whether it conforms to the grammar of the language and then creates an abstract syntax tree. Each interior node of the syntax tree represent a [K]AML programming language construct and each of the nodes represents a component of the construct. This abstract syntax tree is then passed to the interpreter. The interpretation of the program begins at this stage where the tree is walked and the symbol tables are checked in order to resolve variables and the types. Here expression evaluation and type checking takes place. For any invalid arguments, it will throw appropriate exceptions. The symbol table stores the variables of array and integer entries while the function table stores the user-defined functions. Thus, the complete source program is interpreted and the corresponding output is generated.

5.3 Who Implemented What

Kaori implemented the lexer, part of interpreter and made a few edits to the grammar.

Ankush implemented the ast, part of the interpreter and made a few edits to the grammar.

Maninder implemented the grammar, the printer and part of the interpreter.

Mayur implemented the grammar, symbol and function table and part of the interpreter.

6. Test Plan

The Test Plan aims to test the entire Interpreter. All tests can be executed with the command make test. The testing framework basically includes a Makefile and set of various test cases which are the source files with extension **.kaml**.

6.1 Test Suites

The following are the set of tests that we used to test the functionality of the language. These tests represent the successful execution of the test cases.

File	Description
arrayMultiAssign1.kaml	To assign a 2D array to a new array.
arrayMultiAssign2.kaml	To assign selected rows of a 2D array to a new array.
arrayMultiInitialize.kaml	Initialize a 2D array.
arrayNumberMulti.kaml	Access a particular index of a 2D array.
arrayNumberSingle.kaml	Access a particular index of a 1D array.
arraySingleAssign1.kaml	To assign a 1D array to a new array.
arraySingleAssign2.kaml	To assign selected rows of a 1D array to a new array.
arraySingleInitialize.kaml	Initialize a 1D array.
differenceArrayMulti1.kaml	Set Difference operation on two 2D arrays.
differenceArrayMulti2.kaml	Set Difference operation on two 2D arrays, case with duplicates removed.
differenceArrayMulti3.kaml	Set Difference operation on two 2D arrays using selection.
differenceArraySingle1.kaml	Set Difference operation on two 1D arrays,
differenceArraySingle2.kaml	Set Difference operation on two 1D arrays, case with duplicates removed.
differenceArraySingle3.kaml	Set Difference operation on two 1D arrays using range.
differenceArraySingle4.kaml	Set Difference operation on two 1D arrays using selection.
intersectArrayMulti1.kaml	Set Intersect operation on two 2D arrays.
intersectArrayMulti2.kaml	Set Intersect operation on two 2D arrays, case with duplicates removed.
intersectArrayMulti3.kaml	Set Intersect operation on two 2D arrays using selection.

intersectArraySingle1.kaml	Set Intersect operation on two 1D arrays.
intersectArraySingle2.kaml	Set Intersect operation on two 1D arrays, case with duplicates removed.
intersectArraySingle3.kaml	Set Intersect operation on two 1D arrays using range.
intersectArraySingle4.kaml	Set Intersect operation on two 1D arrays using selection.
matrix_multiply.kaml	To Multiply by two 2D matrices.
minusMulti.kaml	To Reverse a 2D array and assign it to a new array.
minusMulti2.kaml	To Reverse a 2D array and assign its selected rows to a new array using range.
minusMulti3.kaml	To Reverse a 2D array and assign its selected rows to a new array using range as well as selection.
minusSingle.kaml	To Reverse a 1D array and assign it to a new array
minusSingle2.kaml	To Reverse a 1D array and assign its selected rows to a new array using range and selection.
simpleAssign.kaml	To assign a constant value to a variable.
unionArrayMulti1.kaml	Set Union operation on two 2D arrays.
unionArrayMulti2.kaml	Set Union operation on two 2D arrays, case with duplicates removed.
unionArrayMulti3.kaml	Set Union operation on two 2D arrays using selection.
unionArrayMulti4.kaml	Set Union operation on two 2D arrays using range.
unionArraySingle1.kaml	Set Union operation on two 1D arrays.
unionArraySingle2.kaml	Set Union operation on two 1D arrays, case with duplicates removed.
unionArraySingle3.kaml	Set Union operation on two 1D arrays using range.
unionArraySingle4.kaml	Set Union operation on two 1D arrays using selection.
determinant.kaml	To calculate Determinant of n*n matrix using recursion.
test_arith_div.kaml	Test the arithmetic division.
test_arith_sum.kaml	Test the arithmetic sum.
test_arith_sub.kaml	Test the arithmetic subtract.
test_arith_product.kaml	Test the arithmetic product.
test_arith_sum_prod_div.kaml	Test the sum, product and division.
test_array_ret_func.kaml	Test the function that returns arrays.
test_array_transpose.kaml	Transpose of a Matrix.

test_delete.kaml	Test the delete operation in arrays.
test_fib.kaml	Test for Fibonacci series.
test_for1.kaml	Test the for loop.
test_hash.kaml	Test the hash expression.
test_insert.kaml	Test the insert expression.
test_mutual_functioncalls.kaml	Calling between two functions.
test_plus.kaml	Test the plus expression.
test_concat.kaml	Test the concatenate expression.
test_cond.kaml	Test the condition operation.

The following are the set of tests that we used to test the exception handling of the language. These tests represent the failure of execution of the test cases.

File	Description
farrayMultiInitialize.kaml	Column Dimension Mismatch In 2D Initialization
farrayMultiInitialize1.kaml	Invalid Column Dimension during 2D Initialization
fdiffArray1.kaml	Difference operation not possible between 1D and 2D arrays
fdiffArrayMulti1.kaml	No of columns should be same in difference operation
fintersectArray1.kaml	Intersect operation not possible between 1D and 2D arrays
fintersectArrayMulti1.kaml	No of columns should be same in intersect operation
fintersectArrayMulti2.kaml	Nothing should be specified in column index during set operations
fminusSingle.kaml	Exception Handling for undeclared indentifier
fminusSingle1.kaml	Exception Handling if RHS is not an array
fsimpleArrayAssign.kaml	Invalid Single Array Initialize Assignment
funionArray.kaml	Invalid Parameters in Set Operation
funionArray1.kaml	Union operation not possible between 1D and 2D arrays
funionArrayMulti1.kaml	No of columns should be same in union operation
fdivideByZero,kaml	Divide by zero
funionArrayMulti2.kaml	Nothing should be specified in column index during set operations

6.2 Test cases for Success

6.2.1 To assign a 2D array to a new array

```
// arrayMultiAssign1.kaml
  [][2]b = {{42,43},{50,60}};
  [][2]a = [][]b;
  for(i=0;i<#[]a;i++)
  {
          show("\n");
          for(j=0;j<(#[][]a/#[]a);j++)
          {
                  show([i][j]a);

          }
  }
```

The corresponding output is:

```
42 43
50 60
```

6.2.2 To assign selected rows of a 2D array to a new array.

```
// arrayMultiAssign2.kaml
  [][2]b = {{42,43},{50,60},{1,2},{4,3},{1,1}};
  [][2]a = [0..2,4][]b;
  for(i=0;i<#[]a;i++)
  {
          show("\n");
          for(j=0;j<(#[][]a/#[]a);j++)
          {
                  show([i][j]a);

          }
  }
```

The corresponding output is:

```
42 43
50 60
1 2
1 1
```

6.2.3 To initialize a 2D array

```
// arrayMultiInitialize.kaml
  [][2]a = {{42,43},{50,60}};
  for(i=0;i<#[]a;i++)
  {
          show("\n");
          for(j=0;j<(#[][]a/#[]a);j++)
          {
                  show([i][j]a);

          }
  }
```

The corresponding output is:

```
42 43
50 60
```

6.2.4 To access a particular index of a 2D array

```
// arrayNumberMulti.kaml
  [][2]a={{42,43},{60,76}};
  b = [1][1]a;
  show(b);
```

The corresponding output is:

```
76
```

6.2.5 To access a particular index of a 1D array

```
// arrayNumberSingle.kaml
  []a={42,43,60,76};
  b = [2]a;
  show(b);
```

The corresponding output is:

```
60
```

6.2.6 To assign a 1D array to a new array

```
// arraySingleAssign1.kaml
  []b = {42,43,50,60};
  []a = []b;
  for(i=0;i<#[]a;i++)
  {
          show([i]a);
  }
```

The corresponding output is:

```
42 43 50 60
```

6.2.7 To assign selected rows of a 1D array to a new array

```
// arraySingleAssign2.kaml
  []b = {42,43,50,60};
  []a = [0..2]b;
  for(i=0;i<#[]a;i++)
  {
          show([i]a);
  }
```

The corresponding output is:

```
42 43 50
```

6.2.8 To initialize a 1D array

```
// arraySingleInitialize.kaml
  []a = {42,43,50,60};
  for(i=0;i<#[]a;i++)
  {
          show([i]a);
  }
```

The corresponding output is:

```
42 43 50 60
```

6.2.9 Set Intersect operation on two 2D arrays

```
// intersectArrayMulti1.kaml
  [][2]a = {{42,43},{50,60},{12,2}};
  [][2]b = {{42,41},{50,60},{12,2}};
  [][2]c = %=([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
50 60
12 2
```

6.2.10 Set Intersect operation on two 2D arrays, case with duplicates removed

```
// intersectArrayMulti2.kaml
  [][2]a = {{50,60},{50,60},{12,2}};
  [][2]b = {{50,60},{50,60},{12,2}};
  [][2]c = %=([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
50 60
12 2
```

6.2.11 Set Intersect operation on two 2D arrays using selection

```
// intersectArrayMulti3.kaml
  [][2]a = {{42,43},{50,61},{12,2},{1,2},{3,4}};
  [][2]b = {{42,41},{50,61},{12,2}};
  [][2]c = %=([0..3][]a,[1,2][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
50 61
12 2
```

6.2.12 Set Intersect operation on two 1D arrays

```
// intersectArraySingle1.kaml
  []a = {42,43,50,60};
  []b = {45,43,50,69};
  []c=%=([]a,[]b);
  for(i=0;i<#[]c;i++)
  {
          show([i]c);
  }
```

The corresponding output is:

```
43 50
```

6.2.13 Set Intersect operation on two 1D arrays, case with duplicates removed

```
// intersectArraySingle2.kaml
  []a = {42,43,43,60};
  []b = {45,43,50,42};
  []c=%=([]a,[]b);
  for(i=0;i<#[]c;i++)
 { show([i]c); }
```

The corresponding output is:

```
42 43
```

6.2.14 Set Intersect operation on two 1D arrays using range

```
// intersectArraySingle3.kaml
  []a = {42,43,50,60};
  []b = {45,43,50,69};
  []c=%=([]a,[2..3]b);
  for(i=0;i<#[]c;i++)
  {
          show([i]c);
  }
```

The corresponding output is:

```
50
```

6.2.15 Set Intersect operation on two 1D arrays using selection

```
// intersectArraySingle4.kaml
  []a = {42,43,50,60};
  []b = {45,43,50,69};
  []c=%=([]a,[1,3]b);
  for(i=0;i<#[]c;i++)
  {
          show([i]c);
  }
```

The corresponding output is:

```
43
```

6.2.16 Set difference operation on two 2D arrays

```

  [][2]a = {{42,43},{50,61},{12,2}};
  [][2]b = {{42,41},{50,60},{12,2}};
  [][2]c = %-([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
```

```
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);

        }
}
```

The corresponding output is:

```
42 43
50 61
```

6.2.17 Set difference operation on two 2D arrays, case with duplicates removed

```
[][2]a = {{42,43},{12,2},{12,2},{13,3}};
[][2]b = {{42,41},{50,60},{12,2}};
[][2]c = %-([][]a,[][]b);

for(i=0;i<#[]c;i++)
{
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);

        }
}
```

The corresponding output is:

```
42 43
13 3
```

6.2.18 Set difference operation on two 2D arrays using selection

```
[][2]a = {{42,43},{50,61},{12,2},{1,2},{3,4}};
[][2]b = {{42,41},{50,60},{12,2}};
[][2]c = %-([0..3][]a,[1,2][]b);
for(i=0;i<#[]c;i++)
```

```
{
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);

        }
}
```

The corresponding output is:

```
42 43
50 61
1 2
```

6.2.19 Set difference operation on two 1D arrays

```
[]a = {42,43,50,60};
[]b = {45,43,50,69};
[]c=%-([]a,[]b);
for(i=0;i<#[]c;i++)
{
        show([i]c);
}
```

The corresponding output is:

```
42 60
```

6.2.20 Set difference operation on two 1D arrays, case with duplicates removed

```
[]a = {42,43,50,50};
[]b = {45,43,50,69};
[]c=%-([]a,[]b);
for(i=0;i<#[]c;i++)
{
        show([i]c);
}
```

The corresponding output is:

```
42
```

6.2.21 Set difference operation on two 1D arrays using range

```
[]a = {42,43,50,60};
[]b = {45,43,50,69};
[]c=%-([]a,[0..1]b);
for(i=0;i<#[]c;i++)
{
        show([i]c);
}
```

The corresponding output is:

```
42 50 60
```

6.2.22 Set difference operation on two 1D arrays using selection

```
[]a = {42,43,50,60};
[]b = {45,43,50,69};
[]c=%-([]a,[2,3]b);
for(i=0;i<#[]c;i++)
{
        show([i]c);
}
```

The corresponding output is:

```
42 43 60
```

6.2.23 Set union operation on two 2D arrays

```
[][2]a = {{42,43},{50,60}};
[][2]b = {{42,41},{5,6}};
[][2]c = %+([][]a,[][]b);
for(i=0;i<#[]c;i++)
```

```
{
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);

        }
}
```

The corresponding output is:

```
42 43
50 60
42 41
5 6
```

6.2.24 Set union operation on two 2D arrays, case with duplicates removed

```
[][2]a = {{42,43},{50,60}};
[][2]b = {{42,41},{50,60}};
[][2]c = %+([][]a,[][]b);
for(i=0;i<#[]c;i++)
{
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);

        }
}
```

The corresponding output is:

```
42 43
42 41
50 60
```

6.2.25 Set union operation on two 2D arrays using selection

```
[][2]a = {{42,43},{50,60},{1,2},{3,4}};
[][2]b = {{42,41},{5,6},{1,2}};
[][2]c = %+([][]a,[0,2][]b);
for(i=0;i<#[]c;i++)
{
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);

        }
}
```

The corresponding output is:

```
42 43
50 60
3 4
42 41
1 2
```

6.2.26 Set union operation on two 2D arrays using range

```
[][2]a = {{42,43},{50,60},{1,2},{3,4}};
[][2]b = {{42,41},{5,6},{1,2}};
[][2]c = %+([0..2][]a,[][]b);
for(i=0;i<#[]c;i++)
{
        show("\n");
        for(j=0;j<(#[][]c/#[]c);j++)
        {
                show([i][j]c);
        }
}
```

The corresponding output is:

```
42 43
50 60
42 41
5 6
1 2
```

6.2.27 Set union operation on two 1D arrays

```
[]a = {42,43,50,60};
[]b = {45,41,55,69};
[]c=%+([]a,[]b);
for(i=0;i<#[]c;i++)
{
        show([i]c);
}
```

The corresponding output is:

```
42 43 50 60 45 41 55 69
```

6.2.28 Set union operation on two 1D arrays, case with duplicates removed

```
[]a = {42,43,50,60};
[]b = {45,43,55,60};
[]c=%+([]a,[]b);
for(i=0;i<#[]c;i++)
{
        show([i]c);
}
```

The corresponding output is:

```
42 50 45 43 55 60
```

6.2.29 Set union operation on two 1D arrays using range

```
 []a = {42,43,50,60};
 []b = {45,41,55,69};
 []c=%+([]a,[1,3]b);
 for(i=0;i<#[]c;i++)
 {
         show([i]c);
 }
```

The corresponding output is:

```
42 43 50 60 41 69
```

6.2.30 Set union operation on two 1D arrays using selection

```
 []a = {42,43,50,60};
 []b = {45,41,55,69};
 []c=%+([]a,[0..2]b);
 for(i=0;i<#[]c;i++)
 {
         show([i]c);
 }
```

The corresponding output is:

```
42 43 50 60 45 41 55
```

6.2.31 To reverse a 2D array and assign it to a new array

```
[][2]a={{1,2},{3,4},{5,6}};
[][2]b=-[][]a;
 for(i=0;i<#[]b;i++)
  {
         show("\n");
         for(j=0;j<(#[][]b/#[]b);j++)
         {
                 show([i][j]b);

         }
  }
```

The corresponding output is:

```
5 6
3 4
1 2
```

6.2.32 To reverse a 2D array and assign its selected rows to a new array using range.

```
[][2]a={{1,2},{3,4},{5,6},{1,3},{2,2}};
[][2]b=-[0..2][]a;
 for(i=0;i<#[]b;i++)
  {
          show("\n");
          for(j=0;j<(#[][]b/#[]b);j++)
          {
                  show([i][j]b);

          }
  }
```

The corresponding output is:

```
5 6
3 4
1 2
```

6.2.33 To reverse a 2D array and assign its selected rows to a new array using range as well as selection.

```
[][2]a={{1,2},{3,4},{5,6},{1,3},{2,2}};
[][2]b=-[0..1,2,4][]a;
 for(i=0;i<#[]b;i++)
  {
          show("\n");
          for(j=0;j<(#[][]b/#[]b);j++)
          {
                  show([i][j]b);
          }
  }
```

The corresponding output is:

```
2 2
5 6
3 4
```

```
1 2
```

6.2.34 To reverse a 1D array and assign it to a new array

```
[]a={1,2,3,4,5};
[]b=-[]a;
 for(i=0;i<#[]b;i++)
  {
          show([i]b);

  }
```

The corresponding output is:

```
5 4 3 2 1
```

6.2.35 To reverse a 1D array and assign its selected rows to a new array using range and selection.

```
[]a={1,2,3,4,5,6,7,8};
[]b=-[1..3,5]a;
 for(i=0;i<#[]b;i++)
  {
          show([i]b);

  }
```

The corresponding output is:

```
6 4 3 2
```

6.2.36 To assign a constant value to a variable.

```
  a = 42;
  show(a);
```

The corresponding output is:

```
42
```

6.2.37 Testing the Arithmetic operators.

```
function sumproddiv(a,b,c,d)
{
return a+b*c/d;
}

d = fun sumproddiv(100,3,4,5);

show("d = ", d);
show("\n");

[]x = {42,36,78,100};

show("Array []x is as follows:");
show("\n");
for(j=0;j<#[]x;j++)
{
  show([j]x, " , ");
}

show("\n");


d= fun sumproddiv([0]x,[1]x,[2]x,[3]x);

show("[0]x + [1]x * [2]x / [3]x -> ", d );
show("\n");

d= fun sumproddiv(#[]x,[1]x,+[2..3]x,+[1,3]x);

show("#[]x + [1]x * +[2..3]x / +[1,3]x -> ", d );
show("\n");
```

The corresponding output is:

```
  = 102
Array []x is as follows:
42  , 36  , 78  , 100  ,
[0]x + [1]x * [2]x / [3]x -> 70
#[]x + [1]x * +[2..3]x / +[1,3]x -> 51
```

6.2.38 Testing Array Size Operator

```
]c = {1,2,3,4,5,6};

d = #[]c;

show("Number of elements in array c are : ", d);

show("\n");

//d = #[]e;  Gives error as expected.

[][4]a = {{2,3,4},{1,2,3},{5,6,7,8}};

d = #[][]a;

show("Number of elements in array a are : ", d);

show("\n");

d = #[]a;

show("Number of rows in array a are : ", d);

show("\n");
```

The corresponding output is:

```
Number of elements in array c are : 6
Number of elements in array a are : 12
Number of rows in array a are : 3
```

6.2.39 Testing delete Operation

```
[]x = {1,2,3,4,5,6};

show("#[]x = " , #[]x);
show("\n");

show("[2]x = " , [2]x);
show("\n");
```

```
[]x -> 3;

show("#[]x = " , #[]x);
show("\n");


show("[2]x = " , [2]x);
show("\n");


function deletefirstnelements([]a,n)
{
i=1;
//show("i = ", i,"\n");
for(i=1;i<=n;i++)
{
//show("i = ", i,"\n");
 []a -> i;
}
return []a;
}

function deletelastnelements([]a,n)
{
i=0;
show("#[]a = " , #[]a);
for(i=1;i<=n;i++)
{

  []a -> #[]a;
}
return []a;
}

function arraccept([]d)
{
}

fun arraccept([]x);

[]x = {1,2,3,4,5,6};

[]x = fun deletefirstnelements([]x,2);

show("#[]x = " , #[]x);
show("\n");

show("[2]x = " , [2]x);
show("\n");

show("Testing delete outside function");
show("\n");
[]x = {1,2,3,4,5,6};

show("#[]x = " , #[]x);
show("\n");

show("[5]x = " , [5]x);
show("\n");

show("[0]x = " , [0]x);
show("\n");

[]x -> 6;

show("[4]x = " , [4]x);
show("\n");

show("[0]x = " , [0]x);
show("\n");

show("Testing delete outside function finished");
show("\n");

show("Testing delete outside function");
show("\n");

[]x = {1,2,3,4,5,6};

show("#[]x = " , #[]x);
show("\n");

show("[5]x = " , [5]x);
show("\n");

show("[0]x = " , [0]x);
show("\n");

[]x -> 6;

show("[4]x = " , [4]x);
show("\n");

show("[0]x = " , [0]x);
show("\n");

show("Testing delete outside function finished");
show("\n");



[]x = {1,2,3,4,5,6};

show("Testing for outside function");
show("\n");

show("#[]x = " , #[]x);
show("\n");

show("[5]x = " , [5]x);
show("\n");

show("[0]x = " , [0]x);
show("\n");
i=0;
for(i=0;i<1;i++)
{  []x -> 6;
}
show("[4]x = " , [4]x);
show("\n");

show("[0]x = " , [0]x);
show("\n");

show("Testing for outside function finished");
show("\n");

[]x = {1,2,3,4,5,6};

[]x = fun deletelastnelements([]x,2);

show("#[]x = " , #[]x);
show("\n");


show("[0]x = " , [0]x);
show("\n");
show("[3]x = " , [3]x);
show("\n");


//testing 2-D deletes.
show("TESTING For 2-D arrays");
show("\n");
[][3]y = {{1,2,3},{4,5,6},{7,8,9},{4,2,3},{2,3,4}};

show("#[]y = " , #[]y);
show("\n");

show("#[][]y = " , #[][]y);
show("\n");

show("[0][0]y = " , [0][0]y);
show("\n");

[][]y -> 5;

show("#[]y = " , #[]y);
show("\n");

show("#[][]y = " , #[][]y);
show("\n");

show("[0][0]y = " , [0][0]y);
show("\n");


//[]y -> 4;

show("#[]y = " , #[]y);
show("\n");

show("#[][]y = " , #[][]y);

show("\n");

show("[0][0]y = " , [0][0]y);
show("\n");
```

The corresponding output is:

```
[]x = 6
[2]x = 3
#[]x = 5
[2]x = 4
#[]x = 4
[2]x = 5
Testing delete outside function
#[]x = 6
[5]x = 6
[0]x = 1
[4]x = 5
[0]x = 1
Testing delete outside function finished
Testing delete outside function
#[]x = 6
[5]x = 6
[0]x = 1
[4]x = 5
[0]x = 1
Testing delete outside function finished
Testing for outside function
#[]x = 6
[5]x = 6
[0]x = 1
[4]x = 5
[0]x = 1
Testing for outside function finished
#[]a = 6 #[]x = 4
[0]x = 1
[3]x = 4
TESTING For 2-D arrays
#[]y = 5
#[][]y = 15
[0][0]y = 1
#[]y = 4
#[][]y = 12
[0][0]y = 1
#[]y = 4
#[][]y = 12
[0][0]y = 1
```

6.2.40 Test case calculating nth fibonaci number

```
function fibo(x)
{
?(x < 2)
{
 ret = 1;
}
!
{
 ret1 = fun fibo(x-1);
 ret2 = fun fibo(x-2);
 ret = ret1 + ret2;
}

return ret;
}

fib = fun fibo(5);

show("fib(5) = ",fib,"\n");

fib = fun fibo(10);

show("fib(10) = ",fib,"\n");

fib = fun fibo(20);

show("fib(20) = ",fib,"\n");

fib = fun fibo(1);

show("fib(1) = ",fib,"\n");
```

The corresponding output is:

```
fib(5) = 8
fib(10) = 89
fib(20) = 10946
fib(1) = 1
```

6.2.41 Testing the for loop.

```
a=1;
b=1;

//c=a+b;

for(i=0;i<25;i++)
{
show("a = " ,a, " ");
c = a+b;
show("c = " ,c, " ");
a = b;
show("a = ",a, " ");
b = c;
show("b = ",b, " ");
show("\n");
}

show("outside the loop","\n");
show("a = " ,a, " ");
c = a+b;
show("c = " ,c, " ");
a = b;
show("a = ",a, " ");
b = c+0;
show("b = ",b, " ");
show("\n");
```

The corresponding output is:

```
a = 1  c = 2  a = 1  b = 2
a = 1  c = 3  a = 2  b = 3
a = 2  c = 5  a = 3  b = 5
a = 3  c = 8  a = 5  b = 8
a = 5  c = 13  a = 8  b = 13
a = 8  c = 21  a = 13  b = 21
a = 13  c = 34  a = 21  b = 34
a = 21  c = 55  a = 34  b = 55
a = 34  c = 89  a = 55  b = 89
a = 55  c = 144  a = 89  b = 144
a = 89  c = 233  a = 144  b = 233
a = 144  c = 377  a = 233  b = 377
a = 233  c = 610  a = 377  b = 610
```

```
a = 377  c = 987  a = 610  b = 987
a = 610  c = 1597  a = 987  b = 1597
a = 987  c = 2584  a = 1597  b = 2584
a = 1597  c = 4181  a = 2584  b = 4181
a = 2584  c = 6765  a = 4181  b = 6765
a = 4181  c = 10946  a = 6765  b = 10946
a = 6765  c = 17711  a = 10946  b = 17711
a = 10946  c = 28657  a = 17711  b = 28657
a = 17711  c = 46368  a = 28657  b = 46368
a = 28657  c = 75025  a = 46368  b = 75025
a = 46368  c = 121393  a = 75025  b = 121393
a = 75025  c = 196418  a = 121393  b = 196418
outside the loop
a = 121393  c = 317811  a = 196418  b = 317811
```

6.2.42 Testing the insert operation.

```
]x = {2,3,4,5,6,7,8,9,10};
[][3]y = {{3,4,5},{6,7,8},{10,19,25},{37,48,99},{29,24,26},{22,39,87}};

show("[8]x = ", [8]x);
show("\n");

show("#[]x = ", #[]x);
show("\n");

show("[5][2]y = ", [5][2]y);
show("\n");

show("#[][]y = ", #[][]y);
show("\n");

[]x <- {11,12};

show("[9]x = ", [9]x);
show("\n");

show("[10]x = ", [10]x);
show("\n");

show("#[]x = ", #[]x);
show("\n");

[][]y <- {10,12};
```

```
show("[6][0]y = ", [6][0]y);
show("\n");

x = 6;

//[]x <- {2,4};

//show("If u r here then trouble","\n");

[]y <- {2,3,4,8};

show("#[][]y = ", #[][]y);
show("\n");
```

The corresponding output is:

```
[8]x = 10
#[]x = 9
[5][2]y = 87
#[][]y = 18
[9]x = 11
[10]x = 12
#[]x = 11
[6][0]y = 10
[6][1]y = 12
[6][2]y = 0
#[][]y = 21
[7][0]y = 43
[7][1]y = 10
[7][2]y = 0
#[][]y = 24
```

6.2.43 Testing the plus operation.

```
[]c = {1,2,3,4,5,6};

d = +[0..5]c;

show("Sum of elements in array c is : ", d);

show("\n");
```

```
//d = +[]e;  Gives error as expected.

[][4]a = {{2,3,4},{1,2,3},{5,6,7,8}};

d = +[][]a;

show("Sum of elements in array a is : ", d);

show("\n");

d = +[][3]a;

show("Sum of elements of 4th column in array a are : ", d);

show("\n");

d = +[][2]a;

show("Sum of elements of 3rd column in array a are : ", d);

show("\n");

d = +[0][]a;

show("Sum of elements of 1st row in array a are : ", d);

show("\n");

d = +[2][]a;

show("Sum of elements of 3rd row in array a are : ", d);

show("\n");

//d = +[3][]a;

//show("Sum of elements of 4th row in array a are : ", d); //as expected gives error

//show("\n");
```

The corresponding output is:

```
Sum of elements in array c is : 21
Sum of elements in array a is : 41
Sum of elements of 4th column in array a are : 8
Sum of elements of 3rd column in array a are : 14
Sum of elements of 1st row in array a are : 9
Sum of elements of 3rd row in array a are : 26
```

6.2.44 Testing the mutual function calls.

```
function foo(c,d)
{
?(c > 0)
{
 show(c+d);
fun bar(c-1,d);
}
!
{
 show(c+d);
}
//fun foo (c,d);

}

function bar(e,f)
{
fun foo(e,f);
}

[]a = {1,2,3,4,5,6};
[][2]b = {{6,7},{8,9}};

fun foo([1]a,[1][1]b);
fun foo(1,2);
fun foo([2]a,[3]a);
fun bar(2,3);
fun foo([3]a + [1][1]b , [0][0]b);

show("\n");

//foo([]a,[][]b);
```

The corresponding output is:

```
11 10 9 3 2 7 6 5 4 5 4 3 19 18 17 16 15 14 13 12 11 10 9 8 7 6
```

6.2.45 Testing the conditional operator.

```
?(a < b)
{
show("a<b  ","a= ", a, "\n" );
}
!
{
show("b<a  ","b= ", b, "\n" );
}

?(a > b)
{
show("a>b  ","a= ", a, "\n" );
}
!
{
show("b<a  ","b= ", b, "\n" );
}

?(a < b)
{
show("Only a<b  ","a= ", a, "\n" );
}


?(a > b)
{
show("Only a > b  ","a= ", a, "\n" );
}
```

The corresponding output is:

```
b<a  b= 8
a>b  a= 9
Only a > b  a= 9
```

6.3 Test cases for Failure

6.3.1 Column Dimension Mismatch in 2D Initialization

```
// farrayMultiInitialize.kaml
  [][]a = {{42,43},{50,60}};
  for(i=0;i<#[]a;i++)
  {
          show("\n");
          for(j=0;j<(#[][]a/#[]a);j++)
          {
                  show([i][j]a);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("Column Dimension Mismatch In Initialization")
```

6.3.2 Invalid Column Dimension during 2D Initialization

```
// farrayMultiInitialize1.kaml
  [][-5]a = {{42,43},{50,60}};
  for(i=0;i<#[]a;i++)
  {
          show("\n");
          for(j=0;j<(#[][]a/#[]a);j++)
          {
                  show([i][j]a);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("Invalid Column Dimension ")
```

6.3.3 Difference operation not possible between 1D and 2D arrays

```
// fdiffArray1.kaml
  []a = {42,43,50,60};
  [][2]b = {{42,41},{5,6}};
  [][2]c = %-([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("Difference operation not possible between 1D and 2D
arrays")
```

6.3.4 No of columns should be same in difference operation

```
// fdiffArrayMulti1.kaml
  [][2]a = {{42,43},{50,60}};
  [][3]b = {{42,41,1},{5,6,2}};
  [][2]c = %-([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("No of columns should be same in difference operation")
```

6.3.5 Intersect operation not possible between 1D and 2D arrays

```
// fintersectArray1.kaml
  []a = {42,43,50,60};
  [][2]b = {{42,41},{5,6}};
  [][2]c = %=([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("Intersect operation not possible between 1D and 2D
arrays")
```

6.3.6 No of columns should be same in intersect operation

```
// fintersectArrayMulti1.kaml
  [][2]a = {{42,43},{50,60}};
  [][3]b = {{42,41,1},{5,6,2}};
  [][2]c = %=([][]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("No of columns should be same in intersect operation")
```

6.3.7 Nothing should be specified in column index during set operations

```
// fintersectArrayMulti2.kaml
  [][2]a = {{42,43},{50,60}};
  [][2]b = {{42,41},{5,6}};
  [][2]c = %=([][]a,[][1]b);
  for(i=0;i<#[]c;i++)
  {
          show("\n");
          for(j=0;j<(#[][]c/#[]c);j++)
          {
                  show([i][j]c);

          }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("Nothing should be specified in column index")
```

6.3.8 Undeclared indentifier

```
// fminusSingle.kaml
[]a={1,2,3,4,5};
[]b=-[]c;
 for(i=0;i<#[]b;i++)
  {
          show([i]b);

  }
```

The corresponding output is:

```
Fatal error: exception Failure("Invalid Minus expression: undeclared indentifier c")
```

6.3.9 Exception Handling if RHS is not an array

```
// fminusSingle1.kaml
a=1;
[]b=-[]a;
 for(i=0;i<#[]b;i++)
  {    show([i]b);   }
```

The corresponding output is:

```
Fatal error: exception Failure("Invalid Minus expression: a is not an array")
```

6.3.10 Invalid Single Array Initialize Assignment

```
// fsimpleArrayAssign.kaml

[2]c={5,6,7,8};
```

The corresponding output is:

```
Fatal error: exception Failure("Invalid Single Array Initialize Assignment")
```

6.3.11 Invalid Parameters in Set Operation

```
// funionArray.kaml
  []a = {42,43,50,60};
  [][2]b = {{42,41},{5,6}};
  [][2]c = %+([]a,[][]b);
  for(i=0;i<#[]c;i++)
  {
     show("\n");
     for(j=0;j<(#[][]c/#[]c);j++)
    {
       show([i][j]c);
     }
  }
```

The corresponding output is:

```
Fatal error: exception Failure("Invalid Parameters in Set Operation")
```

6.3.12 Union operation not possible between 1D and 2D arrays

```
// funionArray1.kaml
  []a = {42,43,50,60};
  [][2]b = {{42,41},{5,6}};
  [][2]c = %+([][]a,[][]b);
  for(i=0;i<#[]c;i++)
```

```
 {
     show("\n");
     for(j=0;j<(#[][]c/#[]c);j++)
     {
         show([i][j]c);

     }
 }
```

The corresponding output is:

```
Fatal error: exception Failure("Union operation not possible between 1D and 2D arrays")
```

6.3.13 No of columns should be same in union operation

```
// funionArrayMulti1.kaml
 [][2]a = {{42,43},{50,60}};
 [][3]b = {{42,41,1},{5,6,2}};
 [][2]c = %+([][]a,[][]b);
 for(i=0;i<#[]c;i++)
 {
     show("\n");
     for(j=0;j<(#[][]c/#[]c);j++)
     {
         show([i][j]c);

     }
 }
```

The corresponding output is:

```
Fatal error: exception Failure("No of columns should be same in union operation")
```

6.3.14 Nothing should be specified in column index during set operations

```
// funionArrayMulti2.kaml
 [][2]a = {{42,43},{50,60}};
 [][2]b = {{42,41},{5,6}};
 [][2]c = %+([][]a,[][1]b);
 for(i=0;i<#[]c;i++)
 {
     show("\n");
     for(j=0;j<(#[][]c/#[]c);j++)
```

```
        {
                show([i][j]c);
        }
 }
```

The corresponding output is:

```
Fatal error: exception Failure("Nothing should be specified in column index")
```

6.3.15 Testing the divide for division by zero.

```
function div(a,b)
{
return (a/b);
}

fun div(10,0);
```

The corresponding output is:

```
Fatal error: exception Failure("Error Divide by zero")
```

6.4 Choice of Test Cases

Test cases were selected to test each and every line of the grammar. Appropriate tests were written so that each function is tested with all possible inputs and parameters based on the grammar rules. Checks were made to ensure that each function returned result as expected. Testing invalid arguments also played a major role to make sure that all the possible exceptions are handled and appropriate failure message is given.

6.5 Who Did What

Mayur did the initial testing of the entire grammar and reported bugs to the appropriate team member to resolve them. Ankush and Maninder successfully wrote the test scripts and implemented all the test cases. Kaori and Mayur verified that all the test cases are in order.

7. Lessons Learned

7.1 Kaori Fukuoka

It is important to spend time to learn the structure of language before implementing the interpreter to have a good start. It would have been much better to know how to code in O'caml at first than learning new programming language and developing a new language at the same time. I found it was very helpful to use Subversion and frequent email communication to keep everyone updated. Even though tasks were distributed among team members, understanding the functions not only the ones you are responsible but also for the other members is essential because there are many cases that you need reference to those parts for implementation. Time management is also a big challenge. It is very important to always keep in mind all the work to do with priority especially when need to work on several team projects and assignments in short time. Making the priority is also significant in software development process because some features cannot be implemented until specific goals are completed.

7.2 Ankush Goel

It was an exciting programming experience that helped me learn a lot of new things. The most demanding part was to leave your comfort zone and code in an entirely new environment, which brings with itself new challenges as well as lot of fun. We realized that, initially, even if things are not falling in right places, if you keep trying everything seems to work out with time. Ocaml taught me that the world is not limited to just java and there are lot of other different programming styles waiting to be explored. Also, this was the first time, I discovered the power of Version Control System and let me confess from now on, I am going to use it in all my projects. Finally, I learned that starting early always makes things easier during the end of semester when lot of unanticipated work just magically appears in front of you and it becomes really difficult to perform upto the level you can.

7.3 Maninder Singh

As per my experience Parser is the most important part in language design. It should not restrict too much and also not allow un-necessary syntax to pass through. While writing interpreter sometimes we had to make some changes in the parser and when the interpreter size eventually increased we had to accommodate by writing more interpreter code which could have been handled in the parser instead. While writing my module in the interpreter the most important thing I learnt was that I should always keep track of the effect my part will do on the rest of the language and vice versa otherwise bugs are inevitable.

Getting used to a functional language after writing code in C/C++ was not an easy task but eventually I learnt that my code is much succinct. Thinking in terms of OCAML was not easy initially but after practice it became better and actually the OCAML type checking is very favorable in writing a less buggy code.

Last but the most important is the experience of working in a 4 member team and keeping close co-ordination among all the group members.

7.4 Mayur Lodha

This project helped me to enhance my knowledge on compilers. Implementation of this project helped me learn the various phases involved in writing an interpreter. It clarified the concepts of how the data flows between various components involved in the process and how the data is filtered for each successive component. This project also gave me an opportunity to learn O'caml, though it wasn't easy to start with.

Apart from technical aspects, this project taught me the lessons on leadership, code organization and time management. I was assigned the task of being team leader. Being the leader, I learned that a team leader has to allocate and distribute work evenly among all the team members. He has to identify strong point among his team members and accordingly assign them the tasks. This also helps each team member to be comfortable with the group and give his best on the tasks assigned.

I advise the future teams to have some deadlines set up for the entire process to flow smooth and stick to those deadlines. Have regular meeting with the team members to be in sink with the team. Identify and resolve the loopholes while coding. Use some concurrent version systems to have a shared repository among the team members so that each team member has the updated files and can commit and share files easily.

8. Appendix

8.1 Source Code

(* scanner.mll – Author: Kaori *)

```
{ let str = ref ""
  open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf }
| "//"    { singlecomment lexbuf }
| "/*"    { multicomment lexbuf }
| '\"'    { str := ""; string lexbuf }
| ';'     { SEMI }
| '['     { L_BRACKET }
| ']'     { R_BRACKET }
| '{'     { L_BRACE }
| '}'     { R_BRACE }
| '('     { L_PAREN }
| ')'     { R_PAREN }
| '+'     { PLUS }
| '-'     { MINUS }
| '*'     { TIMES }
| '/'     { DIVIDE }
| '='     { ASSIGN }
| "++"    { INCREMENT }
| "--"    { DECREMENT }
| "+="    { PLUSASSIGN }
| "-="    { MINUSASSIGN }
| "*="    { TIMESASSIGN }
| '#'     { HASH }
| ".."    { RANGE }
| ','     { COMMA }
| "<-"    { INSERT }
| "->"    { DELETE }
| '?'     { IF }
| '!'     { ELSE }
| '>'     { GT }
| '<'     { LT }
| ">="    { GEQ }
| "<="    { LEQ }
| "=="    { EQ }
| "!="    { NEQ }
| "&&"    { AND }
| "||"    { OR }
| "%-"    { DIFFERENCE }
| "%+"  { UNION }
| "%="  { INTERSECT }
| "show"   { SHOW }
| "for"      { FOR }
| "function"{ FUNCTION }
| "fun"     {FUN}
| "return" { RETURN }
| ['0'-'9']+ as lxm { LITERAL(int_of_string lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9']* as lxm { ID(lxm) }
| eof { EOF }

and multicomment = parse
  "*/" { token lexbuf }
| _  { multicomment lexbuf }

and singlecomment = parse
  "\n" { token lexbuf }
| _  { singlecomment lexbuf }
| eof {EOF}

and string = parse
| '\"'              { STR !str }
| '\\'              { esc lexbuf }
| _ as ch          { str := !str ^ (String.make 1 ch); string lexbuf }

and esc = parse
| 't'               { str := !str ^ "\t"; string lexbuf }
| 'n'               { str := !str ^ "\n"; string lexbuf }
| '\"'              { str := !str ^ "\""; string lexbuf }
| '\\'              { str := !str ^ "\\"; string lexbuf }
```

(* parser.mly – Author: Mayur, Maninder *)

```
%{ open Ast %}

%token SEMI L_BRACKET R_BRACKET L_PAREN R_PAREN L_BRACE
R_BRACE COMMA RANGE HASH INSERT DELETE
%token PLUS MINUS TIMES DIVIDE AND OR ASSIGN INCREMENT
DECREMENT PLUSASSIGN MINUSASSIGN
%token TIMESASSIGN DIFFERENCE UNION INTERSECT
%token EQ NEQ LT LEQ GT GEQ
%token RETURN IF ELSE FOR SHOW INT FUNCTION FUN
%token <int> LITERAL
%token <string> ID STR
%token EOF
```

```
%nonassoc ELSE
%nonassoc IF
%nonassoc FOR
%nonassoc SHOW
%nonassoc FUNCTION
%nonassoc NOELSE
%nonassoc FUN

%left ASSIGN PLUSASSIGN MINUSASSIGN TIMESASSIGN
%left DIFFERENCE UNION INTERSECTION
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%left AND OR

%start compilationUnit
%type <Ast.compilationUnit> compilationUnit

%%

/* entry point */
compilationUnit:
   statementList { List.rev $1 }

statementList:
  /* nothing */{[]}
| statementList statementAll {$2 :: $1}

/* everything in language is a list of statements and/or functions */
statementAll:
   statement{$1}
 | functionDeclaration  {$1}

/* all the type of statements. Function call is also a statement */
statement:
   arrayAssignmentExpression SEMI {Expression($1)}
 | simpleAssignmentExpression SEMI{Expression($1)}
 | arraySingleInitialiseExpression SEMI {Expression($1)}
 | arrayMultiInitialiseExpression SEMI {Expression($1)}
 | arraySourceExpression SEMI {Expression($1)}
 | IF L_PAREN conditionalExpression R_PAREN L_BRACE statementList R_BRACE
%prec NOELSE {If($3,List.rev $6,[])}
 | IF L_PAREN conditionalExpression R_PAREN L_BRACE statementList R_BRACE
ELSE L_BRACE statementList R_BRACE    {If($3,List.rev $6,List.rev $10)}
```

```
 | FOR L_PAREN simpleAssignmentExpression SEMI conditionalExpression SEMI
loopExpression R_PAREN L_BRACE statementList R_BRACE {For($3,$5,$7,List.rev
$10)}
 | SHOW L_PAREN showExpressionList R_PAREN SEMI {ShowList($3)}
 | functionCall SEMI {$1}
 | functionSimpleAssignment SEMI {$1}
 | functionArrayAssignment SEMI {$1}

showExpressionList:
   showExpression {[$1]}
 | showExpression COMMA showExpressionList {$1::$3}

showExpression:
   parameters { Show($1) }
 | STR { ShowStr($1) }

arrayAssignmentExpression:
   arrayExpression ASSIGN valueExpression {ArrayAssignmentExpression($1,$3)}

valueExpression:
   arrayTypeExpression {$1}
 | number {$1}

/* represents a 1 or 2 dimensional array */
arrayExpression:
   L_BRACKET listExpression R_BRACKET addDimensionExpression ID
{ArrayExpression($2,$4,$5)}

/* represents the array indexes. can be [1],[a],[1,a],[2..5],[1,a,2..b]*/
listExpression:
   /* nothing */{[Noexpr]}
 | elementList {List.rev $1}

/* represents each element in the array */
elementList:
   elements{[$1]}
 | elementList COMMA elements{$3 :: $1}

elements :
   value{$1}
 | value RANGE value{Range($1,$3)}

value:
  SIGN ID {Id($1,$2)}
 | SIGN LITERAL {Literal($1,$2)}
```

```
SIGN:
    {SAdd}
 | PLUS {SAdd}
 | MINUS {SSub}

/* represents second dimension of 2 dimension array */
addDimensionExpression:
  {[]}
| L_BRACKET listExpression R_BRACKET{$2}

/* expressions that returns an array */
arrayTypeExpression:
   arrayExpression{$1}
 | setExpression{$1}
 | MINUS arrayExpression{Minus($2)}

setExpression:
   UNION L_PAREN arrayExpression COMMA arrayExpression R_PAREN
{SetExpression(Union,$3,$5)}
 | INTERSECT L_PAREN arrayExpression COMMA arrayExpression
R_PAREN{SetExpression(Intersect,$3,$5)}
 | DIFFERENCE L_PAREN arrayExpression COMMA arrayExpression
R_PAREN{SetExpression(Difference,$3,$5)}

simpleAssignmentExpression:
   ID ASSIGN numberTypeExpression {SimpleAssignmentExpression($1,$3)}

number:
   numberType {$1}
 | arrayOperation {$1}

numberType:
   value {$1}
 | L_PAREN arithmeticExpression R_PAREN  {$2}

/* expressions that returns a number i.e. not an array */
numberTypeExpression:
   arithmeticExpression {$1}
 | numberType {$1}
 | arrayOperationExpression {$1}

arrayOperation:
   PLUS arrayExpression{PlusExpression($2)}
 | HASH arrayExpression {HashExpression($2)}

/* operations on array that returns a single number */
```

```
arrayOperationExpression:
  arrayOperation {$1}
 | arrayNumber {$1}

/* accessing individual element of an array [2]a, [2][3]a It will always return a number */
arrayNumber:
  L_BRACKET numberTypeExpression R_BRACKET addDimension ID
{ArrayNumber($2,$4,$5)}

addDimension:
  {Noexpr}
| L_BRACKET numberTypeExpression R_BRACKET{$2}

/* will ultimately return a number */
arithmeticExpression:
  numberTypeExpression PLUS numberTypeExpression   {
ArithmeticExpression($1,Add,$3) }
 | numberTypeExpression MINUS numberTypeExpression  {
ArithmeticExpression($1,Sub,$3) }
 | numberTypeExpression TIMES numberTypeExpression  {
ArithmeticExpression($1,Mul,$3) }
 | numberTypeExpression DIVIDE numberTypeExpression {
ArithmeticExpression($1,Div,$3) }

/* initialising a 1 dimension array */
arraySingleInitialiseExpression:
  arrayInitialiseExpression initialiseList
R_BRACE{ArraySingleInitialiseExpression($1,List.rev $2)}

arrayInitialiseExpression:
  arrayExpression ASSIGN L_BRACE {$1}

/* list of anything that is a number */
initialiseList:
  numberTypeExpression{[$1]}
 | initialiseList COMMA numberTypeExpression {$3 :: $1}

/* initialising a 2 dimension array */
arrayMultiInitialiseExpression:
  arrayInitialiseExpression mainInitialiseList
R_BRACE{ArrayMultiInitialiseExpression($1,List.rev $2)}

/* {{},{},{}} */
mainInitialiseList:
  multiList{[$1]}
 | mainInitialiseList COMMA multiList{$3 :: $1}
```

```
multiList:
   L_BRACE initialiseList R_BRACE{List.rev $2}

/* expression that perform operation on array by taking a array and returning the same
array after performing operation */
arraySourceExpression:
   arrayExpression INCREMENT arrayExpression {Concatenate($1,$3)}
 | arrayExpression INSERT L_BRACE initialiseList R_BRACE {Insert($1,List.rev $4)}
 | arrayExpression DELETE numberTypeExpression {Delete($1,$3)}


/* works on any expression that is anumber. #[]a is a number. so this may also come here
*/
conditionalExpression:
/*   numberTypeExpression {$1} */
 | numberTypeExpression LT  numberTypeExpression
{ConditionalExpression($1,Lt,$3)}
 | numberTypeExpression LEQ numberTypeExpression
{ConditionalExpression($1,Leq,$3)}
 | numberTypeExpression GT  numberTypeExpression
{ConditionalExpression($1,Gt,$3)}
 | numberTypeExpression GEQ numberTypeExpression
{ConditionalExpression($1,Geq,$3)}
 | numberTypeExpression EQ  numberTypeExpression
{ConditionalExpression($1,Eq,$3)}
 | numberTypeExpression NEQ numberTypeExpression
{ConditionalExpression($1,Neq,$3)}
 | conditionalExpression AND conditionalExpression
{ConditionalExpression($1,And,$3)}
 | conditionalExpression OR  conditionalExpression {ConditionalExpression($1,Or,$3)}

/* i++ / j-- / ID = NTE   */
loopExpression:
   ID INCREMENT{Increment($1)}
 | ID DECREMENT{Decrement($1)}
 | simpleAssignmentExpression {$1}

functionCall:
  FUN ID  L_PAREN actualparameterizedExpression R_PAREN {FunctionCall($2,$4)}

actualparameterizedExpression:
   /* nothing */{[]}
 | aparameterList {List.rev $1}

aparameterList:
  parameters {[$1]}
 | aparameterList COMMA parameters {$3 :: $1}

parameters:
  value {$1}
 | PLUS arrayExpression{PlusExpression($2)}
 | HASH arrayExpression {HashExpression($2)}
 | arrayTypeExpression{$1}
 | farithmetics {$1}
 | L_PAREN farithmetics R_PAREN {$2}

farithmetics:
  parameters PLUS parameters   { ArithmeticExpression($1,Add,$3) }
 | parameters MINUS parameters  { ArithmeticExpression($1,Sub,$3) }
 | parameters TIMES parameters  { ArithmeticExpression($1,Mul,$3) }
 | parameters DIVIDE parameters { ArithmeticExpression($1,Div,$3) }


formalparameterizedExpression:
  /* nothing */{[]}
 | fparameterList {List.rev $1}

fparameterList:
  fparameters {[$1]}
 | fparameterList COMMA fparameters {$3 :: $1}

fparameters:
  SIGN ID {Id($1,$2)}
 | arrayExpression{$1}

/* all expressions that can be a number or array */
/*expression:
  numberTypeExpression{$1}
 | setExpression{$1}
 | MINUS arrayExpression{Minus($2)}
 */
functionDeclaration:
  FUNCTION ID L_PAREN formalparameterizedExpression R_PAREN L_BRACE
statementList functionClose {FunctionDeclaration($2,$4,List.rev $7,$8)}

/* function can return number or array */
functionClose:
 R_BRACE {Noexpr}
| RETURN parameters SEMI R_BRACE {$2}

functionSimpleAssignment:
  ID ASSIGN functionCall {FunctionSimpleAssignment($1,$3)}

functionArrayAssignment:
  arrayExpression ASSIGN functionCall {FunctionArrayAssignment($1,$3)}
```

(* ast.mli – Author: Ankush *)

```
type operator = Add | Sub | Mul | Div
type conditionalOperator = Lt | Leq | Gt | Geq | Eq | Neq | And | Or
type setOperator = Union | Intersect | Difference
type sign = SAdd | SSub

type expression =
   ArithmeticExpression of expression * operator * expression
  | ConditionalExpression of expression * conditionalOperator * expression
  | ArrayExpression of expression list * expression list * string
  | ArrayNumber of expression * expression * string
  | Literal of sign * int
  | Id of sign * string
  | Increment of string
  | Decrement of string
  | Minus of expression
  | SimpleAssignmentExpression of string * expression
  | ArrayAssignmentExpression of expression * expression
  | HashExpression of expression
  | PlusExpression of expression
  | SetExpression of setOperator * expression * expression
  | Concatenate of expression * expression
  | Insert of expression * expression list
  | Delete of expression * expression
  | ArraySingleInitialiseExpression of expression * expression list
  | ArrayMultiInitialiseExpression of expression * expression list list
  | Range of expression * expression
  | Noexpr

type statement =
   Expression of expression
  | If of expression * statement list * statement list
  | For of expression * expression * expression * statement list
  | Show of expression
  | ShowStr of string
  | ShowList of statement list
  | FunctionDeclaration of string * expression list * statement list * expression
  | FunctionCall of string * expression list
  | FunctionSimpleAssignment of string * statement
  | FunctionArrayAssignment of expression * statement

type compilationUnit = statement list
```

(* printer.ml – Mayur, Maninder *)

```
open Ast

let string_of_arith_operator = function
    Add -> "+"
  | Sub -> "-"
  | Mul -> "*"
  | Div -> "/"

let string_of_cond_operator = function
    Lt  -> "<"
  | Leq -> "<="
  | Gt  -> ">"
  | Geq -> ">="
  | Eq  -> "="
  | Neq -> "!="
  | And -> "&&"
  | Or  -> "||"

let string_of_set_operator = function
    Union  -> "%+"
  | Intersect  -> "%="
  | Difference -> "%-"

let string_of_signop = function
    SAdd -> "+"
  | SSub -> "-"

let rec string_of_expression = function
    ArithmeticExpression(e1,a_op,e2) ->
          string_of_expression e1 ^ " " ^ string_of_arith_operator a_op ^ " " ^
          string_of_expression e2
  | ConditionalExpression(e1,c_op,e2) ->
          string_of_expression e1 ^ " " ^ string_of_cond_operator c_op ^ " " ^
          string_of_expression e2
  | ArrayExpression (el1,el2,s) ->
          "[" ^ String.concat ", " (List.map string_of_expression el1) ^ "]["
           ^  String.concat ", " (List.map string_of_expression el2) ^ "]" ^ s
  | ArrayNumber (e1,e2,s) ->
                    "[" ^ string_of_expression e1 ^ "][" ^ string_of_expression e2 ^ "]" ^ s
  | Literal(sign_op, lit) -> string_of_signop sign_op ^ string_of_int lit
  | Id(sign_op, str) -> string_of_signop sign_op ^ str
  | Increment(s) -> s ^ " ++"
  | Decrement(s) -> s ^ " --"
  | Minus(e) -> "- " ^ string_of_expression e
  | Range(e1,e2) ->
          string_of_expression e1 ^ " .. " ^ string_of_expression e2
  | SimpleAssignmentExpression(str,e) -> str ^ " = " ^ string_of_expression e
  | ArrayAssignmentExpression(e1,e2) ->
          string_of_expression e1 ^ " = " ^ string_of_expression e2
  | HashExpression(e) -> "# " ^ string_of_expression e
  | PlusExpression(e) -> "+ " ^ string_of_expression e
  | SetExpression(s_op,e1,e2) ->
          string_of_set_operator s_op ^ "( " ^ string_of_expression e1 ^ " , " ^
          string_of_expression e2 ^ " )\n"
  | Concatenate(e1,e2) ->
          string_of_expression e1 ^ "++" ^ string_of_expression e2
  | Insert(e1,e2) -> string_of_expression e1 ^ " <- " ^ "(InitializeList)\n"
  | Delete(e1,e2) ->
          string_of_expression e1 ^ " -> " ^ string_of_expression e2
  | ArraySingleInitialiseExpression(e,el) ->
          string_of_expression e ^ " = " ^  "ArraySingleInitialiseExpression."
  | ArrayMultiInitialiseExpression(e,ell) ->
          string_of_expression e ^ " = " ^  "ArrayMultiInitialiseExpression."
  | Noexpr -> ""


let rec string_of_statement = function
    Expression(e) -> string_of_expression e
  | If(e,sl1,sl2) ->
          "if(" ^ string_of_expression e ^ ")\n{\n" ^ String.concat "\n" (List.map
          string_of_statement sl1) ^                  "\n}\nelse\n{\n" ^ String.concat "\n"
          (List.map string_of_statement sl2) ^ "\n}\n"
  | For(e1,e2,e3,s) ->
          "for(" ^ string_of_expression e1 ^ " ; " ^ string_of_expression e2 ^ " ; " ^
          string_of_expression e3 ^ ")\n{\n" ^ String.concat "\n" (List.map
          string_of_statement s) ^ "\n}\n"
  | Show(e) -> string_of_expression e
  | FunctionDeclaration (s, el,sl,e) ->
          "function " ^ s ^ "(" ^  String.concat "," (List.map string_of_expression el) ^ ")" ^
          "{\n" ^
          String.concat "\n" (List.map string_of_statement sl) ^ string_of_expression e ^
          "\n}\n"
  | FunctionCall (s, el) ->
          s ^ "(" ^  String.concat "," (List.map string_of_expression el) ^ ");"
  | FunctionSimpleAssignment(s,stmt) -> s ^ " = " ^ "function call"
  | FunctionArrayAssignment(expr,stmt) -> "[] = function call"
  | ShowStr(s) -> s
  | ShowList(stmt_l) ->
          "show(" ^ String.concat "," (List.map string_of_statement stmt_l) ^  ");"

let string_of_compilationUnit (statements)  =
    String.concat "\n" (List.map string_of_statement statements)
```

(* interpreter.ml – Author: Mayur, Maninder, Ankush, Kaori *)

```
open Ast

module StringHash = Hashtbl.Make(struct
type t = string
let equal x y = x = y
let hash = Hashtbl.hash
end);;

let run (stmlist) =
let ftable = StringHash.create 50
in
 let call stmtlist stable =
    let rec eval env = function
      Noexpr -> -1, env
    (* Number values *)
    | Literal(op,x) ->
        (match op with
              SAdd -> x
        | SSub -> -x), env
    (* Variable names *)
    | Id(op,var) ->
          let v = StringHash.mem env var in
       if v then
          let (size,list)= StringHash.find env var in
         let value = List.hd (snd (size,list)) in
          (match op with
               SAdd -> value
           | SSub -> -value), env
       else raise (Failure ("parameter '" ^ var ^ "' not found"))
    (* Calculate arithmetic expression and returns the result *)
    | ArithmeticExpression(e1, op, e2) ->
        let v1,env = eval env e1 in
         let v2,env = eval env e2 in
          (match op with
             Add -> v1 + v2
        | Sub -> v1 - v2
        | Mul -> v1 * v2
        | Div -> v1 / v2),env
(* Conditional Expression returns 1 if true, 0 otherwise *)
| ConditionalExpression(e1, op, e2) ->
   let v1,env = eval env e1 in
     let v2,env = eval env e2 in
       (match op with
           Lt  -> if v1 <  v2 then 1 else 0
         | Leq -> if v1 <= v2 then 1 else 0
         | Gt  -> if v1 >  v2 then 1 else 0
         | Geq -> if v1 >= v2 then 1 else 0
         | Eq  -> if v1 =  v2 then 1 else 0
         | Neq -> if v1 != v2 then 1 else 0
         | And -> if v1 = 1 && v2 = 1 then 1 else 0
         | Or  -> if v1 = 1 || v2 = 1 then 1 else 0),env
(* Increment values by 1: a++ *)
| Increment(var) ->
       let v = StringHash.mem env var in
   if v then
    let (size,list)= StringHash.find env var in
         let v1 = List.hd list in
          let v2 = v1 + 1 in
      (StringHash.replace env var ([],[v2])); v2, env
   else raise (Failure ("Can not call function with parameter " ^ var))
(* Decrement values by 1: a-- *)
| Decrement(var) ->
       let v = StringHash.mem env var in
       if v then
        let (size,list)= StringHash.find env var in
      let v1 = List.hd list in
       let v2 = v1 - 1 in
      (StringHash.replace env var ([],[v2])); v2, env
   else raise (Failure ("Can not call function with parameter " ^ var))
(* Assignment expression for numbers: a=2; - returns assigned value *)
| SimpleAssignmentExpression(var1,e) ->
   let v1,env = eval env e in
          (match e with
           Id(op,var) -> let (size,list)= StringHash.find env var in
                   let rsize = fst (size,list) in
                      if List.length rsize != 0 then
       raise (Failure ("LHS is a number type but RHS is not a number type"))
       else (StringHash.replace env var1 ([],[v1]))
         | _ -> (StringHash.replace env var1 ([],[v1]))); v1,env
(* array initialization for single dimension *)
| ArraySingleInitialiseExpression(arr,list) ->
```

```
        let dim = (match arr with
                ArrayExpression(size1,size2,var) ->
                        (match (size1,size2) with
                                ([Noexpr],[]) -> 1
                            | ([Noexpr],[Noexpr]) -> 2
                            | _ -> 0)
            | _ -> 0 ) in
        if dim = 0 then
                raise (Failure ("Invalid Single Array Initialize Assignment"))
        else
        if dim = 1 then
      let key, env = str env arr 0 in
       let vlist = List.map (fun a -> fst a) (List.map (eval env) list) in
        (StringHash.replace env key ([1;List.length vlist],vlist));0,env
        else raise (Failure ("Invalid Array Assignment Expression: dimensions of LHS
        and RHS do not match"))
(* array initialization for two dimension *)
| ArrayMultiInitialiseExpression(arr,list) ->
    let key, env = str env arr 0 in
          let (size,list1) = StringHash.find env key in
                let csize =  List.hd (List.rev size) in
                  let vlist2 = if csize<0 then
                                raise (Failure ("Invalid Column Dimension "))
                           else List.map (List.map (eval env)) list in
                  let vlist = List.map (List.map (fun a -> fst a)) vlist2 in
                    let _ = List.map (fun a -> if List.length a > csize then
                 raise (Failure ("Column Dimension Mismatch In Initialization"))
                else 0) vlist in
                 let list2 = List.map (fun a -> a @ (Array.to_list (Array.make (if
                (List.length a<csize) then csize-(List.length a) else (List.length a) mod
                csize) 0))) vlist in
                 let templist = List.concat list2 in
                   let rsize = if csize==0 then 1
                        else ((List.length templist) / csize) in
                        (StringHash.replace env key ([rsize;csize],templist));0,env
(* Assign values to the existing array *)
| ArrayAssignmentExpression(arr,e1) ->
     (match e1 with
        Minus(arr1) ->
                 let key, env = str env arr1 1 in
                   let v = StringHash.mem env key in
                           if v then
                    let (size,list1) = StringHash.find env key in
                    match size with
                [] -> raise (Failure ("Invalid Minus expression: " ^ key ^ " is not an
                array"))
```

```
           | _ ->
            let rlist = if StringHash.mem env "temp&" then
                         (snd (StringHash.find env "temp&")) else [] in
             let _ = StringHash.remove env "temp&" in
              let list = if List.length rlist = 0 then list1
               else
             if List.hd size = 1 then
              (List.map (fun i -> List.nth list1 i) rlist)
             else getSelectedRows rlist size list1 in
               let mkey, env = str env arr 0 in
            ( if List.hd size = 1 then
                    (StringHash.replace env mkey ([1;List.length list],List.rev list))
            else
              let finallist =
                let rec reverse rlist tlist s e =
                  let temparr = Array.of_list rlist in
                    let temp2 = Array.sub temparr s e in
                    let jlist = (tlist @ Array.to_list temp2) in
                  if s > 0 then reverse rlist jlist (s-e) e else jlist in
                       let revlist = [] in
                      let eindex =  (List.hd (List.tl size)) in
                      let sindex = (((List.length list/eindex)-1)*(eindex)+0) in
                       reverse list revlist sindex eindex in
                      (StringHash.replace env mkey ([(List.length finallist)/(List.hd
                      (List.tl size));List.hd (List.tl size)],finallist)))
            else raise (Failure ("Invalid Minus expression: undeclared indentifier " ^
           key))
  (* array set operation: Union, Intersect and Difference *)
   | SetExpression(op,a1,a2) ->
       let key1, env = str env a1 1 in
       let flag1 = StringHash.mem env key1 in
           let f1 = (match a1 with
                 ArrayExpression(size1,size2,var) ->
                    (match (size1,size2) with
                          ([Noexpr],[]) -> 1
                        | ([Noexpr],[Noexpr]) -> 2
                        | _ -> 0)
               | _ -> 0                ) in
           let f2 = (match a2 with
                ArrayExpression(size1,size2,var) ->
                     (match (size1,size2) with
                          ([Noexpr],[]) -> 1
                         | ([Noexpr],[Noexpr]) -> 2
                         | _ -> 0)
              | _ -> 0                 ) in
           let (size1,list) = if flag1==false then
```

```
                    raise (Failure ("Undeclared Identifier " ^ key1))
            else if (f1==1 && f2==2) || (f1==2 && f2==1) then
                    raise (Failure ("Invalid Parameters in Set Operation"))
            else StringHash.find env key1 in
             let rlist = if StringHash.mem env "temp&" then
                    (snd (StringHash.find env "temp&")) else [] in
          let _ = StringHash.remove env "temp&" in
           let list1 = if List.length rlist = 0 then list
            else
            if List.hd size1 = 1 then (List.map (fun i -> List.nth list i) rlist)
             else getSelectedRows rlist size1 list in
            let key2, env = str env a2 1 in
              let flag2 = StringHash.mem env key2 in
              let (size2,list) = if flag2==false then
                    raise (Failure ("Undeclared Identifier " ^ key2))

                    else StringHash.find env key2 in
     let rlist = if StringHash.mem env "temp&" then
                    (snd (StringHash.find env "temp&")) else [] in
             let _ = StringHash.remove env "temp&" in
              let list2 = if List.length rlist = 0 then list
               else
              if List.hd size2 = 1 then (List.map (fun i -> List.nth list i) rlist)
              else getSelectedRows rlist size2 list in
              let csize1 =  List.hd (List.rev size1) in
                let csize2 = List.hd (List.rev size2) in
                  let rsize1 = List.hd size1 in
                    let rsize2 = List.hd size2 in
                      let mkey, env = str env arr 0 in
                       (match op with
            (* Set Operation – Union *)
            Union -> if rsize1 == 1 && rsize2 == 1 then
                    let d = list1 @ list2 in
                    let unionlist =
                     let rec lst d =
                     match d with
                     [] -> []
                    | [x] -> [x]
                    | x :: remainder ->
                    if (List.mem x remainder) then lst (remainder)
                    else [x] @ lst (remainder) in
                     lst d in (StringHash.replace env mkey ([1;csize1],unionlist))
                    else
                    (if rsize1>1 && rsize2>1 then if csize1!=csize2 then raise (Failure
                    ("No of columns should be same in union operation"))
                    else(
                    let dim1 = (match a1 with
                     ArrayExpression(size1,size2,var) ->
                            let v1,env = eval env (List.hd size2) in
                            if v1 != -1 then 1 else 0
                      |_ -> -1)                                in
                    let dim2 =  (match a2 with
                    ArrayExpression(size1,size2,var) ->
                    let v1,env = eval env (List.hd size2) in
                         if v1 != -1 then 1 else 0
                            |_-> -1                                ) in
                    if dim1 == 1 || dim2 ==1 then
                    raise (Failure ("Nothing should be specified in column index"))
                      else
                    let totallist = list1 @ list2 in
                       let ulist =
                           let rec ftemp totallist =
                                if (List.length totallist)<(2*csize1) then [totallist]
                                else (let templst1 = Array.to_list (Array.sub (Array.of_list
                                totallist) 0 csize1) in
                                let x = [templst1] in
                                let z = Array.to_list (Array.sub (Array.of_list totallist)
                                csize1 ((List.length totallist) - csize1))
                            in x @ ftemp z) in
                                   ftemp totallist in
                                let finallist =
                                 let rec xyz tmp =
                                        (match tmp with
                                          [] -> []
                                        | [x] -> [x]
                                        | x :: remainder ->
                                       if (List.mem x remainder) then xyz (remainder)
                                        else [x] @ xyz (remainder)) in
                                       xyz ulist in (StringHash.replace env mkey
                                ([(List.length finallist);csize1],(List.concat finallist))))
                                       else (raise (Failure ("Union operation not possible
                                between 1D and 2D arrays"))))
(* Set Operation – Intersect *)
 | Intersect ->
                     if rsize1 == 1 && rsize2 == 1 then
                        let intersectlist =
                           let rec inter list1 list2=
                                 match list1 with
                                          [] -> []
                                       | [x] -> if (List.mem x list2) then [x] else []
                                       | x :: remainder ->
                                       if (List.mem x list2) then [x] @ inter remainder list2
                        else inter remainder list2 in
                              inter list1 list2 in
                              let intersectfinallist =
                  let rec lst d =
                   match d with
                    [] -> []
         | [x] -> [x]
         | x :: remainder ->
                  if (List.mem x remainder) then lst (remainder)
       else [x] @ lst (remainder) in
         lst intersectlist in
                  (StringHash.replace env mkey
      ([1;csize1],intersectfinallist))
       else
  (if rsize1>1 && rsize2>1 then if csize1!=csize2 then
raise (Failure ("No of columns should be same in intersect
operation"))
else(
      let dim1 = (match a1 with
   ArrayExpression(size1,size2,var) ->
      let v1,env = eval env (List.hd size2) in
                  if v1 != -1 then 1 else 0
   |_ -> -1)                                   in
      let dim2 =  (match a2 with
        ArrayExpression(size1,size2,var) ->
      let v1,env = eval env (List.hd size2) in
                  if v1 != -1 then 1 else 0
  |_-> -1                                             ) in
      if dim1 == 1 || dim2 ==1 then
      raise (Failure ("Nothing should be specified in column
index")) else
let ilist1 =
       let rec ftemp1 totallist1 =
      if (List.length totallist1)<(2*csize1) then [totallist1]
else (let templst1 = Array.to_list (Array.sub (Array.of_list
totallist1) 0 csize1) in

let x = [templst1] in
let z = Array.to_list(Array.sub(Array.of_list
totallist1)csize1((List.length totallist1)-csize1))
  in x @ ftemp1 z) in
 ftemp1 list1 in
  let ilist2 =
   let rec ftemp2 totallist2 =
       if (List.length totallist2)<(2*csize1) then [totallist2]
```

```
                        else (let templst2 = Array.to_list (Array.sub (Array.of_list
                        totallist2) 0 csize1) in
                  let x = [templst2] in
                 let z = Array.to_list(Array.sub(Array.of_list
                 totallist2)csize1((List.length totallist2)-csize1))
                 in x @ ftemp2 z) in
                  ftemp2 list2 in
                  let finallist =
                    let rec ixyz itmp1 itmp2 =
                      match itmp1 with
                             [] -> []
                  | [x] -> if (List.mem x itmp2) then [x] else []
                  | x :: remainder -> if (List.mem x itmp2) then [x] @ ixyz
                 remainder itmp2                         else ixyz remainder itmp2 in
                                ixyz ilist1 ilist2 in
                                let intersectfinallist =
                                let rec xyz tmp =
                                 (match tmp with
                                     [] -> []
                                   | [x] -> [x]
                                   | x :: remainder ->
                                if (List.mem x remainder) then xyz (remainder)
                            else [x] @ xyz (remainder)) in
                                xyz finallist in (StringHash.replace env mkey
                        ([(List.length intersectfinallist);csize1],(List.concat
                        intersectfinallist))))
                                else (raise (Failure ("Intersect operation not possible
                        between 1D and 2D arrays"))))

          (* Set Operation – Difference *)
| Difference -> if rsize1 == 1 && rsize2 == 1 then
                        let difflist =
                         let rec diff list1 list2 =
                          match list1 with
                            [] -> []
                         | [x] -> if (List.mem x list2) then [] else [x]
                                  | x :: remainder -> if (List.mem x list2) then
                                diff remainder list2
                                  else [x] @ diff remainder list2 in
                                diff list1 list2 in (StringHash.replace env mkey
                        ([1;csize1],difflist))
                          else
                        (if rsize1>1 && rsize2>1 then if csize1!=csize2 then raise
                        (Failure ("No of columns should be same in difference
                        operation"))
                        else(
        let dim1 = (match a1 with
      ArrayExpression(size1,size2,var) ->
            let v1,env = eval env (List.hd size2) in
                  if v1 != -1 then 1 else 0
   |_ -> -1)                                in
     let dim2 =  (match a2 with
 ArrayExpression(size1,size2,var) ->
            let v1,env = eval env (List.hd size2) in
                  if v1 != -1 then 1 else 0
   |_-> -1                                         ) in
        if dim1 == 1 || dim2 ==1 then
        raise (Failure ("Nothing should be specified in column
        index")) else
        let dlist1 =
         let rec ftemp1 totallist1 =
        if (List.length totallist1)<(2*csize1) then [totallist1]
                else(let templst1 = Array.to_list (Array.sub
        (Array.of_list totallist1) 0 csize1) in
        let x = [templst1] in
        let z = Array.to_list (Array.sub(Array.of_list
        totallist1)csize1((List.length totallist1)-csize1))
        in x @ ftemp1 z) in
                  ftemp1 list1 in
        let dlist2 =
          let rec ftemp2 totallist2 =
                if (List.length totallist2)<(2*csize1) then [totallist2]
                else(let templst2 = Array.to_list (Array.sub
        (Array.of_list totallist2) 0 csize1) in                  let x =
        [templst2] in
        let z = Array.to_list(Array.sub(Array.of_list
        totallist2)csize1((List.length totallist2)-csize1))
        in x @ ftemp2 z) in
                  ftemp2 list2 in
        let finallist =
         let rec dxyz dtmp1 dtmp2 =
                match dtmp1 with
                      [] -> []
          | [x] -> if (List.mem x dtmp2) then [] else [x]
          | x :: remainder -> if (List.mem x dtmp2) then dxyz
        remainder dtmp2
        else [x] @ dxyz remainder dtmp2 in
         dxyz dlist1 dlist2 in
         (StringHash.replace env mkey ([(List.length
        finallist);csize1],(List.concat finallist))))
         else (raise (Failure ("Difference operation not possible
        between 1D and 2D arrays")))))
     | ArrayExpression(size1,size2,var) ->
       let rhsdim =
         if (StringHash.mem env var) then
         let rows = List.hd (fst (StringHash.find env var)) in
              (match rows with
                   0 -> 0
                  | 1 -> 1
                  | _ -> 2
                          )
             else
         raise (Failure ("The Variable "  ^ var ^ " not found for array assignment "
         )) in
             let lhsdim =
              (match arr with
                    ArrayExpression(size3,size4,var2) ->
                            if (StringHash.mem env var2) then
                            let rows = List.hd (fst (StringHash.find env var2)) in
                            (match rows with
                              0 ->0
          | 1 -> 1
          | _ -> 2
                 )
        else
         (match (size3,size4) with
            ([Noexpr],[Noexpr]) -> 2
              |  ([Noexpr],[]) -> 1
          |  ([Noexpr],hd :: tl) -> 2
                            | _    ->
                            raise (Failure ("Wrong expression on Lhs for array
                            assignment.")))
                    |  _ -> raise (Failure ("Wrong expression on Lhs for array
         assignment."))
                        ) in
let test  =    ( match (lhsdim,rhsdim) with
                 (1,2) -> 3
               | (2,1) -> 4
               | _  ->  -2
                        ) in
if test = 3 then
   (match (size1,size2) with
      ([Noexpr],y::[]) -> (match arr with
                    ArrayExpression(size3,size4,var2) ->
              let (size,list) = StringHash.find env var in
                  let dm_y , _ = eval env y in
                   let columns = List.hd(List.rev size) in
                    if dm_y >= columns || dm_y < 0 then
                   raise (Failure ("Invalid arguments in array " ^ var ))
              else
                let columnlist =
                  let rec f lst n last =
                    if n >= last then
                     lst
                       else f ([(List.nth list n)]@lst) (n+columns) last
                    in f [] dm_y (List.length list)
                        in (StringHash.replace env var2 ([1]@[List.length
                       columnlist],(List.rev columnlist)))
                | _ -> raise(Failure ("Wrong expression on Lhs for array
       assignment."))
                          )
| (x::[],[Noexpr]) -> (match arr with
                       ArrayExpression(size3,size4,var2) ->
        let (size,list) = StringHash.find env var in
          let dm_x , _ = eval env x in
            let rows = List.hd size in
             if dm_x >= rows || dm_x < 0 then
                    raise (Failure ("Invalid arguments in array " ^ var ))
            else
              let rowlist =
                let rec f lst n last =
                  if n >= last then
                    lst
             else f ([(List.nth list n)]@lst) (n+1) last

                               in f [] (dm_x*(List.hd(List.rev size)))
                               ((dm_x+1)*(List.hd(List.rev size)))
                               in (StringHash.replace env var2 ([1]@[List.length
                               rowlist],(List.rev rowlist)))
                       | _ -> raise(Failure ("Wrong expression on Lhs for array
              assignment."))
                                        )
              |  _  ->  raise(Failure ("Wrong expression on Rhs for array
              assignment."))
                           )
       else if test = 4 then
     (match arr with
         ArrayExpression(size3,size4,var2) ->
            if StringHash.mem env var2 then
         (match (size3,size4) with
             ([Noexpr],y::[]) ->
            let (size,list) = StringHash.find env var2 in
              let dm_y , _ = eval env y in
                let rows = List.hd size in
          let columns = List.hd (List.rev size) in
        if dm_y >= columns || dm_y < 0 then
                    raise (Failure ("Invalid arguments in array " ^ var2 ))
         else
          let (sz,lst) = StringHash.find env var in
         let colR = List.hd (List.rev sz) in
         if colR = rows then
         let arr1 = Array.of_list list in
         let rec f ar n step ls =
         (match ls with
                  []   -> ar
               |  hd::tl -> let _ = ar.(n) <- hd in f ar (n+step) step tl          )
       in let _ = f arr1 dm_y columns lst in
           let newlst = Array.to_list arr1 in
         StringHash.replace env var2 (size,newlst)
         else
         raise (Failure ("Wrong expression on Lhs for array assignment."))
| (x::[],[Noexpr]) ->
let (size,list) = StringHash.find env var2 in
let dm_y = List.hd (List.rev(size)) in
let dm_x , _ = eval env x in
let rows = List.hd size in
   if dm_x >= rows || dm_x < 0 then
raise (Failure ("Invalid arguments in array " ^ var2 ))
  else
 let (sz,lst) = StringHash.find env var in
let colR = List.hd (List.rev sz) in
if colR = dm_y then
 let arr1 = Array.of_list list in
let rec f ar n ls =
         (match ls with
            []   -> ar
           |  hd::tl -> let _ = ar.(n) <- hd in f ar (n+1) tl                     )
 in let _ = f arr1 (dm_x*colR) lst in
  let newlst = Array.to_list arr1 in
              StringHash.replace env var2 (size,newlst)
  else
         raise (Failure ("Wrong expression on Lhs for array assignment."))
         |  _  -> raise (Failure ("Wrong expression on Lhs for array
         assignment."))                                              )
         else
         raise (Failure ("Array expression on Lhs for array assignment is
         not defined"))
          | _ -> raise (Failure ("Wrong expression on Lhs for array
         assignment."))                                       )
         else
```

```
              (let arrdim_used =
                 (match e1 with
                    ArrayExpression(size1,size2,var) ->
                       (match (size1,size2) with
                                   ([Noexpr],[Noexpr]) -> 22
                                   | ([Noexpr],_) -> 1
                                   | (_,[Noexpr]) -> 2
                  | _ -> let v1 = List.length size2 in
                                   if v1 != 1 then 1 else 0  )
                  | _ -> 0
                     ) in
       if arrdim_used!=0 then
           let key, env = str env e1 1 in
             let (size,list) = StringHash.find env key in
               let rlist = if StringHash.mem env "temp&" then
                          (snd (StringHash.find env "temp&")) else [] in
              let _ = StringHash.remove env "temp&" in
                 let mlist = if List.length rlist = 0 then list
                       else
                          if List.hd size = 1 then (List.map (fun i -> List.nth list i)
               rlist)
                          else getSelectedRows rlist size list in
        let mkey, env = str env arr 0 in
          let eindex = List.hd (List.tl size) in
             if List.hd size = 1 then
               StringHash.replace env mkey ([1;List.length mlist],mlist))
               else
               (StringHash.replace env mkey ([List.length
    mlist/eindex;eindex],mlist))
               else
                          let value,env = eval env e1 in
                    (match arr with
                     ArrayExpression(ex1,ex2,var) ->
                      let (size,list) = StringHash.find env var in
                         let arr = Array.of_list list in
                          let v1,env = eval env (List.hd ex1) in
                          if List.length ex2 = 0 then
                            Array.set arr v1 value
                           else
                          (let v2,env = eval env (List.hd ex2) in
                           let cindex = List.hd (List.rev size) in
                           Array .set arr ((v1*cindex)+v2) value);
                          let list1 = Array.to_list arr in
                           (StringHash.replace env var (size,list1))
               | _ -> ()) )
    | _ -> let value,env = eval env e1 in
```

```
                              (match arr with
                               ArrayExpression(ex1,ex2,var) ->
                                let (size,list) = StringHash.find env var in
                                  let arr = Array.of_list list in
                                         let v1,env = eval env (List.hd ex1) in
                                         if List.length ex2 = 0 then
                                Array.set arr v1 value
                                         else
                                         (let v2,env = eval env (List.hd ex2) in
                                     let cindex = List.hd (List.rev size) in
                                       Array .set arr ((v1*cindex)+v2) value);
                                                   let list1 = Array.to_list arr in
                                                    (StringHash.replace env var (size,list1))
                                 | _ -> ())
                               );0,env
  (* Hash returns the length of an array: #[]a *)
  | HashExpression(arr) ->
         let key, env = str env arr 1 in
         let v = StringHash.mem env key in
       if v then
           let (size,list) = StringHash.find env key in
                     match size with
                              [] -> raise (Failure ("Invalid Hash expression: " ^ key ^ " is not an
                    array"))
                             | _ -> let length =
                    (match arr with
           ArrayExpression(size1,size2,var) ->
             (match (size1,size2) with
                       ([Noexpr],[]) ->
                              if ((List.hd size) == 1) then List.length list
                              else ((List.length list)/(List.hd (List.rev size)))
             | ([Noexpr],[Noexpr]) -> List.length list
             | _ -> 0  )
         | _ -> 0
                     )in length, env
                    else raise (Failure ("Invalid Hash expression: cannot find array " ^ key))
   (* Plus return the sum of the elements in an array: +[]a *)
   | PlusExpression(arr) ->
      let key, env = str env arr 1 in
       let v = StringHash.mem env key in
       if v then
        let dim = (match arr with
            ArrayExpression(size1,size2,var) ->
              (match size2 with
                          [] -> 1
               | _  -> 2 )
```

```
                    | _ -> raise (Failure ("Invalid expression send in Plus expression"))

          ) in
    if dim = 2 then
      (match arr with
        ArrayExpression(size1,size2,var) ->
          (match (size1, size2) with
                  ([Noexpr],[Noexpr]) ->
                            let (size,list) = StringHash.find env key in
                            let sum = List.fold_left (fun f s -> f + s) 0 list in sum, env
          | ( x :: [],[Noexpr])  ->
                            let (size,list) = StringHash.find env key in
             let dm_x , _ = eval env x in
             let rows = List.hd size in
              if dm_x >= rows || dm_x < 0 then
            raise (Failure ("Invalid arguments in array " ^ var ))
              else
              let rowlist =
              let rec f lst n last =
               if n >= last then
                 lst
               else f ([(List.nth list n)]@lst) (n+1) last
                      in f [] (dm_x*(List.hd(List.rev size))) ((dm_x+1)*(List.hd(List.rev
             size)))
                      in let sum = List.fold_left (fun f s -> f + s) 0 rowlist in sum, env
           | ([Noexpr],y::[]) ->
             let (size,list) = StringHash.find env key in
             let dm_y , _ = eval env y in
             let columns = List.hd(List.rev size) in
             if dm_y >= columns || dm_y < 0 then
            raise (Failure ("Invalid arguments in array " ^ var ))
             else
              let columnlist =
              let rec f lst n last =
               if n >= last then
                 lst
               else f ([(List.nth list n)]@lst) (n+columns) last
                 in f [] dm_y (List.length list)
                            in let sum = List.fold_left (fun f s -> f + s) 0 columnlist in
                            sum, env
             |  _ -> raise (Failure ("Invalid expression send in Plus expression"))
                      )
         | _ -> raise (Failure ("Invalid expression sent in Plus expression"))
          )
             else
               let (size,list) = StringHash.find env key in
```

```
                    match size with
                    [] -> raise (Failure ("Invalid Plus expression: " ^ key ^ " is not an array"))
                  |  _ -> let rlist = if StringHash.mem env "temp&" then
                              (snd (StringHash.find env "temp&")) else [] in
                         let _ = StringHash.remove env "temp&" in
                              let mlist = if List.length rlist = 0 then list else (List.map (fun i ->
                    List.nth list i) rlist) in
                         let sum = List.fold_left (fun f s -> f + s) 0 mlist in sum, env
                                else raise (Failure ("Invalid Plus expression: cannot find array " ^
                              key))
(* Concatenate two arrays and stores in the first array: []a++[]b *)
| Concatenate(array1, array2) ->
  let key1, env = str env array1 1 in
  let key2, env = str env array2 1 in
  let v1 = StringHash.mem env key1 in
   if v1 then
    let v2 = StringHash.mem env key2 in
     if v2 then
     let (size1, list1) = StringHash.find env key1 in
     let (size2, list2) = StringHash.find env key2 in
        match (size1, size2) with
                    [], _ -> raise (Failure ("Invalid Concatenate expression: " ^ key1 ^ " is not
                    an array"))
                    |_, [] -> raise (Failure ("Invalid Concatenate expression: " ^ key2 ^ " is not
                    an array"))
        | _ ->
                    let rlist = if StringHash.mem env "temp&" then (snd (StringHash.find env
                    "temp&")) else [] in
                     let _ = StringHash.remove env "temp&" in
                     let clist2 = if List.length rlist = 0 then list2 else
                              (List.map (fun i -> List.nth list2 i) rlist) in
                    let concat = list1 @ clist2 in
          let m,n =
                     if List.hd size1 = 1 && List.hd size2 = 1 then List.hd size1, List.hd
                    (List.tl size1) + List.hd (List.tl size2)
                     else
                              (if List.hd size1 != 1 && List.hd size2 != 1 && List.hd (List.tl
                              size1) = List.hd (List.tl size2) then
                         List.hd size1 + List.hd size2, List.hd (List.tl size1)
                              else raise (Failure ("The dimensions of array " ^ key1 ^ " and array
                              " ^ key2 ^ " do not match")) ) in
                        (StringHash.replace env key1 (m::[n], concat));0,env
                    else raise (Failure ("Invalid concatenation argument: cannot find "^ key2))
                    else raise (Failure ("Invalid concatenation argument: cannot find "^ key1))
| ArrayNumber(e1,e2,name) -> let v1,env = eval env e1 in
      let v2,env = eval env e2 in
    let (size,list) = if StringHash.mem env name then StringHash.find env name
                      else raise (Failure ("Undeclared Identifier " ^ name)) in
      let arr = Array.of_list list in
        if v2 = -1 then arr.(v1),env
        else
          let cindex = List.hd (List.rev size) in
            arr.((v1*cindex)+v2),env
| ArrayExpression(size1,size2,var) ->
  let tlist1 = List.map (fun a -> fst a) (List.map (eval env) size1) in
   let tlist2 = List.map (fun a -> fst a) (List.map (eval env) size2) in
    if List.length tlist1 <= 1 && List.length tlist2 <= 1 then
     let (size,list) = StringHash.find env var in
      let arr = Array.of_list list in
       if List.length tlist2 = 0 then arr.(List.hd tlist1),env
       else
       let cindex = List.hd (List.rev size) in
        arr.((List.hd tlist1*cindex)+List.hd tlist2),env
    else 0,env
 (* Insert new element to the existing array *)
 | Insert(arrtype_e,el) ->
    let var, env = str env arrtype_e 1 in
      let list_var = List.map (fun e -> let v,_ = eval env e in v) el in
        let len = List.length list_var in
          let bl = StringHash.mem env var in
              if bl then
               let rows = List.hd (fst (StringHash.find env var)) in
                let arrdim_used =
                (match arrtype_e with
                  ArrayExpression(size1,size2,var) ->
                  (match (size1,size2) with
                     ([Noexpr],[]) -> 1
                   | ([Noexpr],[Noexpr]) -> 2
                   | _ -> 0  )
               | _ -> raise (Failure ("Invalid expression send in Plus expression"))
               ) in
                let acc_dim =
                (match rows with
                      0 ->0
                    | 1 -> 1
                    | _ -> 2
                ) in
                if arrdim_used = acc_dim then
                 let column_width =
                      List.hd (List.tl (fst (StringHash.find env var))) in
                  let orig_list = snd (StringHash.find env var) in
                  (match rows with
                    1 -> StringHash.replace env var
                ([1]@[column_width+len],orig_list@list_var)
              | _ -> let rows2add = len / column_width in
                 let modu =  len mod column_width in
                   if modu = 0 then
                         StringHash.replace env var
                ([rows+rows2add]@[column_width],orig_list@list_var)
                   else
                    let padded_lst =
                     let rec f s el =
                      (match s with
                        0 -> el
                   | _ -> f (s-1) el@[0] ) in f (column_width - modu) list_var in
                StringHash.replace env var
                ([rows+rows2add+1]@[column_width],orig_list@padded_lst)
                        )
             else
                raise (Failure ("Wrong array used in Insert expression."));0,env
              else
                raise (Failure ("Unknown array used in Insert expression."))
| Delete(array1, e) ->
   let key1, env = str env array1 1 in
    let v1,env = eval env e in
                let arrdim_used =

                  (match array1 with
           ArrayExpression(size1,size2,var) ->
             (match (size1,size2) with
                   ([Noexpr],[]) -> 1
                  | ([Noexpr],[Noexpr]) -> 2
                  | _ -> 0  )
               | _ -> 0
               ) in
                 if v1<=0 then
                 raise (Failure ("Invalid index for the element to be deleted"))
                  else(
                        let (size1, list1) = if StringHash.mem env key1 then
                        StringHash.find env key1 else
                        raise (Failure ("Undeclared Identifier " ^ key1)) in
                         if v1>List.length list1 then
                        raise (Failure ("Invalid index for the element/row to be deleted"))
                                   else( let csize1 =  List.hd (List.rev size1) in
                                  let rsize1 = List.hd size1 in

                         if rsize1==1 then
                        let _=  if arrdim_used == 2 then
                        raise (Failure ("Array is 1 Dimension but here used like 2
                Dimension")) in
                   (let finallist =
                   if v1==1 then List.tl list1
                else if v1==(List.length list1) then List.rev (List.tl (List.rev list1))
                else
                 let temparr = Array.of_list list1 in
                 let temp2 = Array.sub temparr 0 (v1-1) in
                 let temp3 = Array.sub temparr v1 ((Array.length temparr) - v1) in
                let temp4 = Array.append temp2 temp3 in
                Array.to_list temp4 in
                        (StringHash.replace env key1 ([1;(List.length finallist)], finallist)))
                        else(
                        let _=  if arrdim_used == 1 then
                        raise (Failure ("Array is 2 Dimension but here used like 1
                Dimension")) in
                         let finallist=
                        if v1==1 then Array.to_list(Array.sub (Array.of_list list1)
                        (csize1*v1) ((List.length list1) - (csize1*v1)))
                        else
                        if v1==((List.length list1)/csize1) then Array.to_list (Array.sub
                        (Array.of_list list1) 0 (csize1*(v1-1)))
                        else
                                   let temparr = Array.of_list list1 in
                        let temp2 = Array.sub temparr 0 (csize1*(v1-1)) in
                        let temp3 = Array.sub temparr (csize1*v1) ((Array.length temparr)
                        - (csize1*v1)) in
                        let temp4 = Array.append temp2 temp3 in
                        Array.to_list temp4 in
                                   (StringHash.replace env key1 ([(rsize1-1);(csize1)],
                        finallist)))));0, env
| Range(x1,x2)->
    let k1, env = eval env x1 in
     let k2, env = eval env x2 in
      let xst = [k1;k2] in
                   if k1>k2 || k1<0 || k2<0 then
                raise (Failure ("Invalid parameters for range"))
                else
                let list =
                   let rec rnge xs =
             (match xs with
              [] -> []
             | [x] -> [x]
             | x :: y -> if x < (List.hd y) then [x] @ rnge [x+1;(List.hd y)]
              else [(List.hd y)])
```

```
                in rnge xst in (StringHash.add env "rangelist&" ([List.length list],list));-
      2,env
| _ -> 0,env

 and str env arr flag =
     match arr with ArrayExpression(size1,size2,var)->
                let tlist1 = List.map (fun a -> fst a) (List.map (eval env) size1)in
         let list1 =
           let rec build1 xs =
                (match xs with
                  [] -> []
                | x :: y -> if x = -2 then if flag = 1 then
                 let lst = snd (StringHash.find env "rangelist&") in
                 let dim_fst = List.hd (fst (StringHash.find env var)) in
                  if dim_fst = 1 then
                    let dim_snd = List.hd (List.rev (fst (StringHash.find env var))) in
                    let last = List.hd (List.rev lst) in
                    if last >= dim_snd then
                   raise (Failure ("Range exceeds the Array Dimentions"))
                  else (snd (StringHash.find env "rangelist&"))@ build1 y
                   else
                      let last = List.hd (List.rev lst) in
                       if last >= dim_fst then
                   raise (Failure ("Range exceeds the Array Dimentions"))
                  else (snd (StringHash.find env "rangelist&"))@ build1 y
                  else (snd (StringHash.find env "rangelist&"))@ build1 y
                else if x = -1 then
                         []
                 else if flag = 1 then
                 let dim_fst = List.hd (fst (StringHash.find env var)) in
                    if dim_fst = 1 then
                    let dim_snd = List.hd (List.rev (fst (StringHash.find env var))) in
                      if x >= dim_snd then
                   raise (Failure ("Range exceeds the Array Dimentions"))
                    else [x]@build1 y
                    else
                     if x >= dim_fst then
                   raise (Failure ("Range exceeds the Array Dimentions"))
                    else [x]@build1 y
                     else [x]@build1 y
                                          )     in build1 tlist1 in
                    let _ =(StringHash.remove env "rangelist&") in
                    let tlist2 = List.map (fun a -> fst a) (List.map (eval env) size2) in
            let list2 =
           let rec build2 xs =
                (match xs with
              [] -> []
             | x :: y -> if x = -2 then if flag = 1 then
        let lst = (snd (StringHash.find env "rangelist&")) in
          let last = List.hd ( List.rev lst) in
               let dim_snd = List.hd (List.rev (fst (StringHash.find env var))) in
             if last >= dim_snd then
              raise (Failure ("Range exceeds the Array Dimentions"))
             else (snd (StringHash.find env "rangelist&"))@ build2 y
             else (snd (StringHash.find env "rangelist&"))@ build2 y
            else if x = -1 then
                  []
               else if flag = 1 then
               let dim_snd = List.hd (List.rev (fst (StringHash.find env var))) in
            if x >= dim_snd then
            raise (Failure ("Range exceeds the Array Dimentions"))
                 else [x]@build2 y
                 else [x]@build2 y
           )
      in build2 tlist2 in let _ = StringHash.remove env "rangelist&" in
        if flag = 0 then
         let list1 = if List.length list1 = 0 then [0] else list1 in
          let list2 = if List.length list2 = 0 then [0] else list2 in
           (StringHash.add env var (list1@list2,[]))
        else
               if List.length list1 >= 1 then (StringHash.add env "temp&" ([],list1))
          else ()
        ;var,env
 | _ -> "",env;

and getSelectedRows rlist size list =
(let rec rnge xs =
  (match xs with
    [] -> []
   | [x] -> [x]
   | x :: y -> if x < (List.hd y) then [x] @ rnge [x+1;(List.hd y)]
             else [(List.hd y)]) in
  let mylist =
    let rec elements rlist tlist eindex =
      match rlist with
        [] -> []
      | x :: y ->  let tlist = tlist@rnge [x;x+eindex-1] in
              if List.length y = 0 then tlist
              else elements y tlist eindex in
      let eindex = List.hd (List.tl size) in
    elements (List.map (fun a -> a * eindex) rlist) [] eindex in
(List.map (fun i -> List.nth list i) mylist));
```

```
in
  let rec exec env = function
    Expression(e) -> let _, env = eval env e in env
  | Show(e) -> let v,env = eval env e in print_string (string_of_int v);print_string " "; env
  | ShowStr(str) -> print_string str; env
  | ShowList(list) -> ignore (List.map (fun i -> exec env i) list); env
  | For(assign,cond,loop,stmtlist) -> let v1,env = eval env assign in
                                          while(let c,env = eval env cond in c = 1) do
                                          ignore (List.fold_left exec env stmtlist);
                                          let _,_ = eval env loop in ()
                                          done; env
  | If(con,stm1,stm2) -> let v, env = eval env con in
                              if v = 1 then List.fold_left exec env stm1 else List.fold_left exec
                    env stm2;
  | FunctionDeclaration (name, el,stml,e) -> StringHash.add ftable name (el,stml,e);env
  | FunctionSimpleAssignment(id,fcall) ->
                    let _ = exec env fcall in
           if (StringHash.mem env "return") then
         let size_l, val_l = StringHash.find env "return" in
                (match size_l with
              [] ->  let test1 = StringHash.mem env id in
                     if test1 then
                     let _ = StringHash.replace env id (size_l, val_l) in
                               StringHash.remove env "return" ; env
                     else
                     let _ = StringHash.add env id (size_l, val_l) in
                              StringHash.remove env "return"; env
            | _  -> raise (Failure ("Wrong return Type"))
                                          )
          else
                    raise (Failure ("Function expected to return something ID"))
  | FunctionArrayAssignment(arrexpr,fcall) -> let _ = exec env fcall in
            let test = StringHash.mem env "return" in
              if test then
                let size_l, val_l = StringHash.find env "return" in
                 let dimention_r =
                  (match size_l with
                     [] -> 0
                   | hd::tail -> if (hd = 1) then 1 else 2
                         ) in
                 let size1,size2,var = (match arrexpr with
                     ArrayExpression(size1,size2,var) -> size1,size2,var
                   |  _ -> raise (Failure ("Unexpected Type"))
                                          ) in
                  let dimention_l =
                     (match (size1,size2) with
                    ([Noexpr],[]) -> 1
                   | ([Noexpr],[Noexpr]) -> 2
                   |  _    -> raise (Failure ("lvalue not appropriate in function assignment"))
                                  ) in
               if (dimention_r = dimention_l) then
                let _ = StringHash.replace env var (size_l, val_l) in
                         StringHash.remove env "return" ; env
               else
                raise (Failure ("Wrong return Type"))
               else
             raise (Failure ("Function expected to return something ID"))
      | FunctionCall(name, el) -> let test = StringHash.mem ftable name in
              if test then
               let formal_l,stm_l,return = StringHash.find ftable name in
              if List.length formal_l = List.length el then
               let temptable_formals = StringHash.create 50 in
                let isarraynumber e =
                 (match e with
                  ArrayExpression(size1,size2,var) ->
                    (match (size1,size2) with
                      (hd::[],[]) ->
                          ( match hd with
                             Range(e1,e2) ->  0, Noexpr
                           | Noexpr -> 0,Noexpr
                           | _ -> 1, ArrayNumber(hd,Noexpr,var)
                                 )
                     | (hd::[],hd1::[]) ->
                          ( match (hd,hd1) with
                             (Range(e1,e2), _ ) ->  0,Noexpr
                           | ( _ ,Range(e3,e4)) ->  0,Noexpr
                           | ( _ ,Noexpr) ->  0,Noexpr
                           | ( Noexpr ,_ ) ->  0,Noexpr
                           | _          -> 1, ArrayNumber(hd,hd1,var)  )
                     | _ -> 0,Noexpr     )
                 | _ -> 0,Noexpr     )
                             in
                        let _ = List.iter2 (fun fe ae ->
                        (match fe with
                        Id(op,var) -> let v = StringHash.mem temptable_formals var in
                          if v then
                        raise (Failure ("Argument " ^ var ^ " Declared in function  " ^
                        name ^ " has more than one instance."))
                        else
                          let value = 1 in
                            let value1 = (match op with
                               SAdd -> value
```

```
        | SSub -> -value) in
            let value2 , _ =   (match ae with
              Id(op,var1) -> eval env ae
            |  Literal(op,x) -> eval env ae
            |  ArithmeticExpression(e1, op, e2) ->
                     (match (e1, e2) with
                     Minus(e), _ ->
                    raise (Failure ("Can not call function with
                    parameters other than ***"))
            | _ , Minus(e)->
                    raise (Failure ("Can not call function with
                    parameters other than ***"))
            | SetExpression(op,exp1,exp2) , _ ->
                    raise (Failure ("Can not call function with
                    parameters other than ***"))
             | _, SetExpression(op,exp1,exp2) ->
                    raise (Failure ("Can not call function with
                    parameters other than ***"))
| ArrayExpression(size1,size2,var),
ArrayExpression(size3,size4,var1) ->
   let test1,arrexpr1 = isarraynumber e1 in
   let test2,arrexpr2 = isarraynumber e2 in
             ( match (test1, test2) with
            1,1 -> let expr =
            ArithmeticExpression( arrexpr1,op,arrexpr2) in
                     eval env expr
            |  _  ->
                    raise (Failure ("Can not call function with
                    parameters other than ***"))  )
| ArrayExpression(size1,size2,var) ,  _ ->
    let test,arrexpr = isarraynumber e1 in
    ( match test with
            1 -> let expr =
                    ArithmeticExpression( arrexpr,op,e2) in
            eval env expr
| _ ->
raise (Failure ("Can not call function with parameters other than
***"))
 )
| _, ArrayExpression(size1,size2,var)  ->
   let test,arrexpr = isarraynumber e2 in
    ( match test with
            1 -> let expr =
            ArithmeticExpression( e1,op,arrexpr) in
            eval env expr
| _ ->
```

```
                     raise (Failure ("Can not call function with parameters other than
               ***"))
                     )
              | _,_ ->  eval env ae                        )
            |  HashExpression(arr) -> eval env ae
            |  PlusExpression(arr) -> eval env ae
            |  ArrayNumber(e1,e2,name) -> eval env ae
            |  ArrayExpression(size1,size2,var) ->
                 let test,arrexpr = isarraynumber ae in
                    if test = 1 then
                      eval env arrexpr
                        else
                           raise (Failure ("Can not call function with parameters other than
               ***"))
            |  _  -> raise (Failure ("Can not define function with parameters other than
***"))
          ) in
           StringHash.add temptable_formals var ([],[value1*value2])
 |  ArrayExpression(size1,size2,var_f)->
        let v = StringHash.mem temptable_formals var_f in
         if v then
         raise (Failure ("Argument " ^ var_f ^ " Declared in function " ^ name ^ " has more
         than one instance."))
         else
          let dimention_f =
           ( match (size1,size2) with
               ([Noexpr],[]) -> 1
             | ([Noexpr],[Noexpr]) -> 2
             |  _    ->
                  raise (Failure ("Wrong argument type in function declaration " ^ name))
           ) in
            let dimention_a =
             ( match ae with
                ArrayExpression(size1, size2,var) ->
                    let rows = List.hd(fst (StringHash.find env var)) in
                      (match rows with
                          0 -> 0
                        | 1 -> 1
                        | _ -> 2  )
              | Minus(ArrayExpression(size1,size2,var)) ->
                          let rows = List.hd(fst (StringHash.find env var)) in
                      (match rows with
                          0 -> 0
                        | 1 -> 1
                        | _ -> 2  )
```

```
    | SetExpression(op, ArrayExpression(size1, size2, var1),
ArrayExpression(size3,size4,var2)) ->
        if(List.hd(fst (StringHash.find env var1)) =
          List.hd(fst (StringHash.find env var2))) then
      let rows = List.hd(fst (StringHash.find env var1)) in
            (match rows with
               0 -> 0
              | 1 -> 1
              | _ -> 2   )
          else
          raise (Failure ("Different Array Size for setoperation arguments in
          function Call " ^ name))
    | _ -> raise (Failure ("Wrong argument type in function Call " ^ name))
    ) in
      if dimention_f = dimention_a then
        ( match ae with
          ArrayExpression(size1, size2,var) ->
                    StringHash.add temptable_formals var_f (StringHash.find env var)
        | Minus(ArrayExpression(size1,size2,var)) ->
          let _ =  StringHash.add temptable_formals var_f (StringHash.find env var)
in
          let e = Minus(ArrayExpression(size1,size2,var_f)) in
                    let _ = eval temptable_formals  e in ();
          | SetExpression(op, ArrayExpression(size1, size2, var1),
          ArrayExpression(size3,size4,var2)) ->
          let _ =  StringHash.add temptable_formals var_f (StringHash.find env
          var1) in                                          let _ =  StringHash.add
          temptable_formals var2 (StringHash.find env var2) in
          let e =  ArrayAssignmentExpression(ArrayExpression(size1,size2,var_f) ,
          SetExpression(op,ArrayExpression(size1,size2,var_f),
          ArrayExpression(size3,size4,var2))) in
          let _ = eval temptable_formals  e in
          StringHash.remove temptable_formals var2
        | _ ->   raise (Failure ("Wrong argument type in function Call " ^ name))    )
      else
          let _ = print_string "hello " in  raise (Failure ("Wrong argument type in
          function declaration " ^ name))
| _ -> raise (Failure ("Can not define function with parameters other than ***\n"))
)) formal_l el in
 let _ = List.fold_left exec temptable_formals stm_l in
    (match return with
  Id(op,var1) -> let val1 , _ = eval temptable_formals return in
                                    StringHash.add env "return" ([],[val1])
|  Literal(op,x) -> let val1,_ = eval temptable_formals return in
                                   StringHash.add env "return" ([],[val1])
|  ArithmeticExpression(e1, op, e2) ->
```

```
                    (match (e1, e2) with
                          Minus(e), _ ->
                                      raise (Failure ("Wrong Parameters to an arithmetic
                              expression."))
                     | _ , Minus(e)->
                                      raise (Failure ("Wrong Parameters to an arithmetic
                              expression."))
                     | SetExpression(op,exp1,exp2) , _ ->
                                      raise (Failure ("Wrong Parameters to an arithmetic
                              expression."))
                     | _, SetExpression(op,exp1,exp2) ->
                                      raise (Failure ("Wrong Parameters to an arithmetic
                              expression."))
                     | ArrayExpression(size1,size2,var), ArrayExpression(size3,size4,var1) ->
                        let test1,arrexpr1 = isarraynumber e1 in
                        let test2,arrexpr2 = isarraynumber e2 in
                          (match (test1, test2) with
                              1,1 -> let expr =
                                      ArithmeticExpression( arrexpr1,op,arrexpr2) in
                                      let val1,_ = eval temptable_formals expr in StringHash.add
                                      env "return" ([],[val1])
                             |  _  -> raise (Failure ("Wrong Parameters to an arithmetic
                     expression."))
                                                  )
                     | ArrayExpression(size1,size2,var) ,  _ ->  let test,arrexpr = isarraynumber
          e1 in
                                ( match test with
                                    1 -> let expr =
                                          ArithmeticExpression( arrexpr,op,e2) in
                                         let val1,_ = eval temptable_formals expr in
                                                   StringHash.add env "return" ([],[val1])
                                  | _ -> raise (Failure ("Wrong Parameters to an arithmetic
expression."))
                                                  )
                       | _, ArrayExpression(size1,size2,var)  ->  let test,arrexpr =
isarraynumber e2 in
                                ( match test with
                                    1 -> let expr =
                                         ArithmeticExpression( e1,op,arrexpr) in
                                          let val1,_ = eval temptable_formals expr in
                                                   StringHash.add env "return" ([],[val1])
                                   | _ -> raise (Failure ("Wrong Parameters to an arithmetic
expression."))
                                                  )
                      | _,_ ->  let val1,_ =
                    eval temptable_formals return in StringHash.add env "return"
            ([],[val1])
                            )
          |  HashExpression(arr) -> let val1,_ =
                    eval temptable_formals return in StringHash.add env "return"
            ([],[val1])
          |  PlusExpression(arr) -> let val1,_ =
                    eval temptable_formals return in StringHash.add env "return"
            ([],[val1])
          |  ArrayNumber(e1,e2,name) -> let val1,_ =
                    eval temptable_formals return in StringHash.add env "return"
            ([],[val1])
          |  ArrayExpression(size1,size2,var) -> let test,arrexpr = isarraynumber return in
                              if test = 1 then
                               let val1,_ = eval temptable_formals return in
                                        StringHash.add env "return" ([],[val1])
                              else
                              let _ = StringHash.add temptable_formals "return"
                              (StringHash.find temptable_formals var) in
                              let e =  ArrayAssignmentExpression(
                              ArrayExpression(size1,size2,"return") , return) in
                                 let _= eval temptable_formals  e in
                              StringHash.add env "return" (StringHash.find
                              temptable_formals "return")
           | Minus(ArrayExpression(size1,size2,var)) ->
                    let _ = eval temptable_formals  return in StringHash.add env
                    "return" (StringHash.find temptable_formals var)
                 | SetExpression(op,ArrayExpression(size1,size2,var1),
                 ArrayExpression(size3,size4,var2)) ->
                  let e =  ArrayAssignmentExpression(
                         ArrayExpression(size1,size2,var1), return) in
                 let _ = eval temptable_formals  e in
                         StringHash.add env "return" (StringHash.find temptable_formals
                         var1)
           |  _  -> ();
                      )
       else
         raise (Failure ("Wrong Number of arguments in function Call to function " ^
name)); env
    else
         raise (Failure ("Function called must be declared first " ^ name))
  in
    List.fold_left exec stable stmlist
  in
    let stable = StringHash.create 50 in
      call stmlist stable
```

(* kaml.ml – Mayur *)

```
let print = false

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let compilationUnit = Parser.compilationUnit Scanner.token lexbuf in
  if print then
    let listing = Printer.string_of_compilationUnit compilationUnit in
    print_string listing
  else
  ignore (Interpreter.run compilationUnit)
```

8.2 Project Log

```
Revision: 1
Author:
Date: 1:22:40 AM, Friday, November 07, 2008
Message:
Initial directory structure.
----
Added : /branches
Added : /tags
Added : /trunk

Revision: 2
Author: maninder.sonu
Date: 2:05:28 AM, Friday, November 07, 2008
Message:
initial commit
----
Added : /trunk/ast.mli

Revision: 3
Author: maninder.sonu
Date: 2:06:32 AM, Friday, November 07, 2008
Message:
initial commit
----
Added : /trunk/interpreter.ml
Added : /trunk/parser.mly
Added : /trunk/scanner.mll

Revision: 4
```

```
Author: maninder.sonu
Date: 12:13:09 PM, Friday, November 07, 2008
Message:
initial commit
----
Added : /trunk/Makefile

Revision: 5
Author: aankushgoel
Date: 12:52:59 PM, Friday, November 07, 2008
Message:
latest unambigous grammer
----
Modified : /trunk/parser.mly

Revision: 6
Author: aankushgoel
Date: 1:07:54 PM, Friday, November 07, 2008
Message:
latest parser from mayur
----
Modified : /trunk/parser.mly

Revision: 7
Author: maninder.sonu
Date: 2:49:16 PM, Saturday, November 08, 2008
Message:
printer file initial commit
----
Added : /trunk/printer.ml

Revision: 8
Author: mayurlodha
Date: 1:02:24 AM, Monday, November 10, 2008
Message:
modified to include Actions for entire grammar
----
Modified : /trunk/ast.mli

Revision: 9
Author: mayurlodha
Date: 1:02:39 AM, Monday, November 10, 2008
Message:
modified to include Actions for entire grammar
----
Modified : /trunk/parser.mly
```

```
Revision: 10
Author: maninder.sonu
Date: 3:56:10 AM, Monday, November 10, 2008
Message:
printer WORKING and tested
----
Modified : /trunk/printer.ml

Revision: 11
Author: maninder.sonu
Date: 3:57:24 AM, Monday, November 10, 2008
Message:
Use this makefile. It will make ur first working kaml. Usage kaml < testfile.kaml
----
Modified : /trunk/Makefile

Revision: 12
Author: maninder.sonu
Date: 4:48:55 AM, Monday, November 10, 2008
Message:
Initial Top level. Only prints the input file as it is if the grammer is accepted.
----
Added : /trunk/kaml.ml

Revision: 13
Author: mayurlodha
Date: 11:13:55 AM, Monday, November 10, 2008
Message:
modifeid to resolve maninders printer error: single initialise list
----
Modified : /trunk/ast.mli

Revision: 14
Author: mayurlodha
Date: 11:14:34 AM, Monday, November 10, 2008
Message:
grammar updated to put in comments and put in order.
----
Modified : /trunk/parser.mly

Revision: 15
Author: maninder.sonu
Date: 3:48:48 PM, Monday, November 10, 2008
Message:
```

```
A small change in Function decl. BODY is enclosed by L-R_BRACES not L-
R_PARANS
----
Modified : /trunk/parser.mly

Revision: 16
Author: maninder.sonu
Date: 6:32:40 PM, Thursday, November 13, 2008
Message:
Changes for accepting SIGNED Numbers
----
Modified : /trunk/ast.mli

Revision: 17
Author: maninder.sonu
Date: 6:34:21 PM, Thursday, November 13, 2008
Message:
Changes for Signed numbers, ArrayNumbers, functiondecl, etc.
----
Modified : /trunk/parser.mly

Revision: 18
Author: maninder.sonu
Date: 6:34:47 PM, Thursday, November 13, 2008
Message:
Changes for Signed numbers, ArrayNumbers, functiondecl, etc.
----
Modified : /trunk/printer.ml

Revision: 19
Author: maninder.sonu
Date: 6:35:44 PM, Thursday, November 13, 2008
Message:
Initial tests passed.
----
Added : /trunk/tests
Added : /trunk/tests/test1.kaml
Added : /trunk/tests/test_array1.kaml
Added : /trunk/tests/test_func.kaml

Revision: 20
Author: mayurlodha
Date: 12:25:12 AM, Friday, November 14, 2008
Message:
Modified If and For to accept statement list instead of statement
----
```

Modified : /trunk/ast.mli

Revision: 21
Author: mayurlodha
Date: 12:25:58 AM, Friday, November 14, 2008
Message:
Modified If and For to accept statement list instead of statement.
Conditional Expression also modified to take nested conditions.

Modified : /trunk/parser.mly

Revision: 22
Author: mayurlodha
Date: 12:26:21 AM, Friday, November 14, 2008
Message:
Modified to Run Interpreter.

Modified : /trunk/kaml.ml

Revision: 23
Author: mayurlodha
Date: 12:29:31 AM, Friday, November 14, 2008
Message:
Includes arithmetic exression, conditional expression, assignment expression, array expression and If implementation. Symbol table concept working for numbers. So numbers can be stored and retrived in/from symbol table. Arrays only tored in symbol table yet.

Modified : /trunk/interpreter.ml

Revision: 24
Author: maninder.sonu
Date: 5:10:19 PM, Saturday, November 15, 2008
Message:
Added option to build either using interpreter or printer.

Modified : /trunk/kaml.ml

Revision: 25
Author: maninder.sonu
Date: 5:12:11 PM, Saturday, November 15, 2008
Message:
loopexpression modified, if without else is possible.

Modified : /trunk/parser.mly

```
Revision: 26
Author: maninder.sonu
Date: 5:12:45 PM, Saturday, November 15, 2008
Message:
modified according to parser
----
Modified : /trunk/printer.ml

Revision: 27
Author: maninder.sonu
Date: 5:15:33 PM, Saturday, November 15, 2008
Message:
arithmetic exp test.
----
Added : /trunk/tests/test_arith1.kaml

Revision: 28
Author: maninder.sonu
Date: 5:16:17 PM, Saturday, November 15, 2008
Message:
modified array test.
----
Modified : /trunk/tests/test_array1.kaml

Revision: 29
Author: maninder.sonu
Date: 5:17:08 PM, Saturday, November 15, 2008
Message:
test for cond expr.
----
Added : /trunk/tests/test_cond.kaml

Revision: 30
Author: maninder.sonu
Date: 5:17:42 PM, Saturday, November 15, 2008
Message:
test for 'for' expr.
----
Added : /trunk/tests/test_for.kaml

Revision: 31
Author: maninder.sonu
Date: 5:18:16 PM, Saturday, November 15, 2008
Message:
test for numeric expr.
----
```

```
Added : /trunk/tests/test_num1.kaml

Revision: 32
Author: maninder.sonu
Date: 5:18:55 PM, Saturday, November 15, 2008
Message:
modifies func test.
----
Modified : /trunk/tests/test_func.kaml

Revision: 33
Author: mayurlodha
Date: 8:16:30 PM, Sunday, November 16, 2008
Message:
Array Expression problem solved
----
Modified : /trunk/interpreter.ml

Revision: 34
Author: kaorisu0611
Date: 2:05:50 AM, Monday, November 17, 2008
Message:
Added Hash, Plus, Minus, Concatenate part
----
Modified : /trunk/interpreter.ml

Revision: 35
Author: mayurlodha
Date: 9:22:42 PM, Monday, November 17, 2008
Message:
Increment and Decrement type changed
----
Modified : /trunk/ast.mli

Revision: 36
Author: mayurlodha
Date: 9:23:04 PM, Monday, November 17, 2008
Message:
code put in correct order
----
Modified : /trunk/interpreter.ml

Revision: 37
Author: mayurlodha
Date: 9:25:29 PM, Monday, November 17, 2008
Message:
```

```
code removed to remove error. Maninder will look at this issue.
----
Modified : /trunk/kaml.ml

Revision: 38
Author: mayurlodha
Date: 12:10:15 PM, Friday, November 21, 2008
Message:
Modified to remove warnings
----
Modified : /trunk/ast.mli

Revision: 39
Author: mayurlodha
Date: 12:10:35 PM, Friday, November 21, 2008
Message:
Modified to remove warnings
----
Modified : /trunk/printer.ml

Revision: 40
Author: mayurlodha
Date: 12:10:59 PM, Friday, November 21, 2008
Message:
Modified to remove warnings and Get expression removed
----
Modified : /trunk/parser.mly

Revision: 41
Author: mayurlodha
Date: 12:11:31 PM, Friday, November 21, 2008
Message:
Modified to make array expression work
----
Modified : /trunk/interpreter.ml

Revision: 42
Author: mayurlodha
Date: 5:16:53 PM, Friday, November 21, 2008
Message:
Array Number done for 1  n 2 dimension arrays
----
Modified : /trunk/interpreter.ml

Revision: 43
Author: mayurlodha
```

Date: 10:01:26 PM, Saturday, November 22, 2008
Message:
Array Assignment Expression and Set Expression done

Modified : /trunk/interpreter.ml

Revision: 44
Author: aankushgoel
Date: 10:42:41 AM, Sunday, November 23, 2008
Message:
Array Multiinitialise and insert done..some problem with delete..I am working on it...

Modified : /trunk/interpreter.ml

Revision: 45
Author: mayurlodha
Date: 9:17:57 PM, Sunday, November 23, 2008
Message:
Committed for Minus Expression 1 n 2D

Modified : /trunk/interpreter.ml

Revision: 46
Author: kaorisu0611
Date: 4:57:16 AM, Tuesday, November 25, 2008
Message:
Concatenation, For loop, increment and decrement and added test case
----
Modified : /trunk/interpreter.ml
Added : /trunk/tests/concat_for.kaml

Revision: 47
Author: aankushgoel
Date: 3:21:08 PM, Tuesday, November 25, 2008
Message:
set expression done committed for both 1D & 2D

Modified : /trunk/interpreter.ml

Revision: 48
Author: mayurlodha
Date: 9:21:17 PM, Tuesday, November 25, 2008
Message:
range and array expression

Modified : /trunk/interpreter.ml

Revision: 49
Author: aankushgoel
Date: 11:43:34 PM, Tuesday, November 25, 2008
Message:
delete done for both 1D n 2D

Modified : /trunk/interpreter.ml

Revision: 50
Author: mayurlodha
Date: 12:50:10 AM, Wednesday, November 26, 2008
Message:
Warnings handled and code for concatenate and Set expressions done for enhance arrays
----
Modified : /trunk/interpreter.ml

Revision: 51
Author: maninder.sonu
Date: 3:10:36 AM, Wednesday, November 26, 2008
Message:
option for building printer/interpreter

Modified : /trunk/kaml.ml

Revision: 52
Author: maninder.sonu
Date: 3:11:02 AM, Wednesday, November 26, 2008
Message:
for comments

Modified : /trunk/scanner.mll

Revision: 53
Author: maninder.sonu
Date: 3:11:45 AM, Wednesday, November 26, 2008
Message:
nothing much

Modified : /trunk/Makefile

Revision: 54
Author: maninder.sonu
Date: 3:12:39 AM, Wednesday, November 26, 2008
Message:
changes for function parameters. See the actual and formal parameters are different now

----
Modified : /trunk/parser.mly

Revision: 55
Author: maninder.sonu
Date: 3:19:36 AM, Wednesday, November 26, 2008
Message:
function call/declaration insert expression added. Not fully tested with this latest version though.

Modified : /trunk/interpreter.ml

Revision: 56
Author: kaorisu0611
Date: 6:32:14 PM, Thursday, November 27, 2008
Message:

Modified : /trunk/ast.mli
Modified : /trunk/interpreter.ml
Modified : /trunk/parser.mly
Modified : /trunk/scanner.mll

Revision: 57
Author: maninder.sonu
Date: 7:11:47 AM, Friday, November 28, 2008
Message:
For accepting arithmetic expressions in functons

Modified : /trunk/parser.mly

Revision: 58
Author: maninder.sonu
Date: 7:12:00 AM, Friday, November 28, 2008
Message:
For accepting arithmetic expressions in functons

Modified : /trunk/interpreter.ml

Revision: 59
Author: mayurlodha
Date: 9:03:36 PM, Sunday, November 30, 2008
Message:
changes made for show expression and array assignment expression
----
Modified : /trunk/interpreter.ml

Revision: 60
Author: mayurlodha
Date: 10:35:27 PM, Sunday, November 30, 2008
Message:
Fix Bug 3

Modified : /trunk/parser.mly

Revision: 61
Author: aankushgoel
Date: 10:40:39 PM, Sunday, November 30, 2008
Message:
enhanced hash expression

Modified : /trunk/interpreter.ml

Revision: 62
Author: aankushgoel
Date: 12:30:42 AM, Tuesday, December 02, 2008
Message:
bug 4 solved and exception handling added for various function.

Modified : /trunk/interpreter.ml

Revision: 63
Author: aankushgoel
Date: 9:50:38 AM, Tuesday, December 02, 2008
Message:
Bug 5 fixed...

Modified : /trunk/interpreter.ml

Revision: 64
Author: kaorisu0611
Date: 12:00:57 AM, Wednesday, December 03, 2008
Message:
Show fixed - upports multiple inputs of both parameters & string, For fixed, exception handling for Id, Incr, Decr
----
Modified : /trunk/ast.mli
Modified : /trunk/interpreter.ml
Modified : /trunk/parser.mly
Modified : /trunk/scanner.mll

Revision: 65

```
Author: mayurlodha
Date: 12:39:33 AM, Wednesday, December 03, 2008
Message:
Fix Bug 6 & 7. Array Assignment Expression enhanced.
----
Modified : /trunk/interpreter.ml

Revision: 66
Author: mayurlodha
Date: 11:17:08 AM, Wednesday, December 03, 2008
Message:
Fix Bug 8
----
Modified : /trunk/scanner.mll

Revision: 67
Author: kaorisu0611
Date: 3:11:15 PM, Wednesday, December 03, 2008
Message:
show fixed
----
Modified : /trunk/interpreter.ml

Revision: 68
Author: aankushgoel
Date: 8:42:43 PM, Wednesday, December 03, 2008
Message:
bug 12 & bug 13 resolved.
----
Modified : /trunk/interpreter.ml

Revision: 69
Author: aankushgoel
Date: 11:33:27 PM, Wednesday, December 03, 2008
Message:
bug 14 fixed..
----
Modified : /trunk/interpreter.ml

Revision: 70
Author: mayurlodha
Date: 12:03:25 AM, Thursday, December 04, 2008
Message:
Empty Array multi initialise removed
----
Modified : /trunk/parser.mly
```

```
Revision: 71
Author: mayurlodha
Date: 12:34:09 AM, Thursday, December 04, 2008
Message:
Warnings removed
----
Modified : /trunk/interpreter.ml

Revision: 72
Author: maninder.sonu
Date: 4:55:00 PM, Thursday, December 04, 2008
Message:
added fun for calling function
----
Modified : /trunk/ast.mli

Revision: 73
Author: maninder.sonu
Date: 4:55:16 PM, Thursday, December 04, 2008
Message:
added fun for calling function
----
Modified : /trunk/scanner.mll

Revision: 74
Author: maninder.sonu
Date: 4:56:15 PM, Thursday, December 04, 2008
Message:
added return type in function call.
----
Modified : /trunk/interpreter.ml

Revision: 75
Author: maninder.sonu
Date: 4:56:26 PM, Thursday, December 04, 2008
Message:
added return type in function call.
----
Modified : /trunk/parser.mly

Revision: 76
Author: kaorisu0611
Date: 12:50:46 AM, Friday, December 05, 2008
Message:
* Exception Handling for Plus, Minus, Hash, Concatenation, ArraySingleInit
```

* division symbol added to scanner.mll
* changed conditionalExpr in parser.mly

Modified : /trunk/interpreter.ml
Modified : /trunk/parser.mly
Modified : /trunk/scanner.mll

Revision: 77
Author: kaorisu0611
Date: 11:43:07 AM, Wednesday, December 17, 2008
Message:
Bug fixed for Plus, Hash, Minus, Concat

Modified : /trunk/interpreter.ml

Revision: 78
Author: kaorisu0611
Date: 12:07:35 PM, Wednesday, December 17, 2008
Message:
Bug fixed for arraySingleInit

Modified : /trunk/interpreter.ml

Revision: 79
Author: kaorisu0611
Date: 12:14:24 PM, Wednesday, December 17, 2008
Message:
Exception msg changed

Modified : /trunk/interpreter.ml

Revision: 80
Author: mayurlodha
Date: 6:15:08 PM, Wednesday, December 17, 2008
Message:
Modified to allow single array entry updation.

Modified : /trunk/parser.mly

Revision: 81
Author: mayurlodha
Date: 6:15:20 PM, Wednesday, December 17, 2008
Message:
Modified to allow single array entry updation.

Modified : /trunk/interpreter.ml

```
Revision: 82
Author: aankushgoel
Date: 6:56:38 PM, Wednesday, December 17, 2008
Message:
Bug 17 solved..
----
Modified : /trunk/interpreter.ml

Revision: 83
Author: maninder.sonu@gmail.com
Date: 7:38:57 PM, Wednesday, December 17, 2008
Message:
latest parser
----
Modified : /trunk/parser.mly

Revision: 84
Author: maninder.sonu@gmail.com
Date: 7:39:34 PM, Wednesday, December 17, 2008
Message:
warnings removed
----
Modified : /trunk/printer.ml

Revision: 85
Author: maninder.sonu@gmail.com
Date: 7:40:06 PM, Wednesday, December 17, 2008
Message:
changes in insert, hash, function call
----
Modified : /trunk/interpreter.ml

Revision: 86
Author: aankushgoel
Date: 11:10:18 PM, Wednesday, December 17, 2008
Message:
bug xx and xx+1 solved..related to delete
----
Modified : /trunk/interpreter.ml

Revision: 87
Author: aankushgoel
Date: 1:17:06 AM, Thursday, December 18, 2008
Message:
bug related to array assignment and array expression solved
```

```
----
Modified : /trunk/interpreter.ml

Revision: 88
Author: mayurlodha
Date: 1:44:31 AM, Thursday, December 18, 2008
Message:
c=a+b to work
----
Modified : /trunk/parser.mly

Revision: 89
Author: aankushgoel
Date: 2:01:41 AM, Thursday, December 18, 2008
Message:
Another array expression related bug fixed
----
Modified : /trunk/interpreter.ml

Revision: 90
Author: aankushgoel
Date: 2:06:34 AM, Thursday, December 18, 2008
Message:
various test cases and working matrix_multiply
----
Added : /trunk/tests/arrayMultiInitialize.kaml
Added : /trunk/tests/arrayMultiInitialize.out
Added : /trunk/tests/arrayNumberMulti.kaml
Added : /trunk/tests/arrayNumberMulti.out
Added : /trunk/tests/arrayNumberSingle.kaml
Added : /trunk/tests/arrayNumberSingle.out
Added : /trunk/tests/arraySingleInitialize.kaml
Added : /trunk/tests/arraySingleInitialize.out
Added : /trunk/tests/matrix_multiply.kaml
Added : /trunk/tests/matrix_multiply.out
Added : /trunk/tests/simpleAssign.kaml
Added : /trunk/tests/simpleAssign.out

Revision: 91
Author: kaorisu0611
Date: 3:32:17 AM, Thursday, December 18, 2008
Message:
* Minor change
* GET removed
----
Modified : /trunk/interpreter.ml
Modified : /trunk/parser.mly
Modified : /trunk/scanner.mll

Revision: 92
Author: maninder.sonu@gmail.com
Date: 5:13:38 AM, Thursday, December 18, 2008
Message:
Plus modified to get sum of nth row or column. Exception handeling in range done and
many minor changes.
----
Modified : /trunk/interpreter.ml

Revision: 93
Author: maninder.sonu@gmail.com
Date: 5:14:46 AM, Thursday, December 18, 2008
Message:
For expression had statements in reverse order. Probably if also has the same issue. Will
update soon.
----
Modified : /trunk/parser.mly

Revision: 94
Author: maninder.sonu@gmail.com
Date: 5:15:22 AM, Thursday, December 18, 2008
Message:
insert test
----
Added : /trunk/tests/test_insert.kaml

Revision: 95
Author: maninder.sonu@gmail.com
Date: 5:15:54 AM, Thursday, December 18, 2008
Message:
delete test
----
Added : /trunk/tests/test_delete.kaml

Revision: 96
Author: maninder.sonu@gmail.com
Date: 5:16:06 AM, Thursday, December 18, 2008
Message:
hash test
----
Added : /trunk/tests/test_hash.kaml

Revision: 97
```

```
Author: maninder.sonu@gmail.com
Date: 5:16:20 AM, Thursday, December 18, 2008
Message:
plus test
----
Added : /trunk/tests/test_plus.kaml

Revision: 98
Author: maninder.sonu@gmail.com
Date: 6:42:59 AM, Thursday, December 18, 2008
Message:
arith sum
----
Added : /trunk/tests/test_arith_sum.kaml

Revision: 99
Author: maninder.sonu@gmail.com
Date: 6:44:26 AM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_arith_sub.kaml

Revision: 100
Author: maninder.sonu@gmail.com
Date: 6:44:38 AM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_arith_sum_prod_div.kaml

Revision: 101
Author: maninder.sonu@gmail.com
Date: 6:44:47 AM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_arith_product.kaml

Revision: 102
Author: maninder.sonu@gmail.com
Date: 6:44:54 AM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_arith_div.kaml
```

```
Revision: 103
Author: maninder.sonu@gmail.com
Date: 6:45:11 AM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_mutual_functioncalls.kaml

Revision: 104
Author: aankushgoel
Date: 12:15:59 PM, Thursday, December 18, 2008
Message:
Bug 23 solved
----
Modified : /trunk/interpreter.ml

Revision: 105
Author: aankushgoel
Date: 12:18:46 PM, Thursday, December 18, 2008
Message:
Various test cases committed..
----
Added : /trunk/tests/differenceArrayMulti1.kaml
Added : /trunk/tests/differenceArrayMulti1.out
Added : /trunk/tests/differenceArrayMulti2.kaml
Added : /trunk/tests/differenceArrayMulti2.out
Added : /trunk/tests/differenceArraySingle1.kaml
Added : /trunk/tests/differenceArraySingle1.out
Added : /trunk/tests/differenceArraySingle2.kaml
Added : /trunk/tests/differenceArraySingle2.out
Added : /trunk/tests/intersectArrayMulti1.kaml
Added : /trunk/tests/intersectArrayMulti1.out
Added : /trunk/tests/intersectArrayMulti2.kaml
Added : /trunk/tests/intersectArrayMulti2.out
Added : /trunk/tests/intersectArraySingle1.kaml
Added : /trunk/tests/intersectArraySingle1.out
Added : /trunk/tests/intersectArraySingle2.kaml
Added : /trunk/tests/intersectArraySingle2.out
Added : /trunk/tests/minusMulti.kaml
Added : /trunk/tests/minusMulti.out
Added : /trunk/tests/minusSingle.kaml
Added : /trunk/tests/minusSingle.out
Added : /trunk/tests/unionArrayMulti1.kaml
Added : /trunk/tests/unionArrayMulti1.out
Added : /trunk/tests/unionArrayMulti2.kaml
Added : /trunk/tests/unionArrayMulti2.out
Added : /trunk/tests/unionArraySingle1.kaml
Added : /trunk/tests/unionArraySingle1.out
Added : /trunk/tests/unionArraySingle2.kaml
Added : /trunk/tests/unionArraySingle2.out

Revision: 106
Author: aankushgoel
Date: 1:08:16 PM, Thursday, December 18, 2008
Message:
exception related to set expression handled..
----
Modified : /trunk/interpreter.ml

Revision: 107
Author: maninder.sonu@gmail.com
Date: 2:38:12 PM, Thursday, December 18, 2008
Message:
I uploades wrong file by mistake.
----
Modified : /trunk/tests/test_mutual_functioncalls.kaml

Revision: 108
Author: maninder.sonu@gmail.com
Date: 3:16:10 PM, Thursday, December 18, 2008
Message:
changes for functions and hash(exceptions)
----
Modified : /trunk/interpreter.ml

Revision: 109
Author: maninder.sonu@gmail.com
Date: 3:16:38 PM, Thursday, December 18, 2008
Message:
changes for if
----
Modified : /trunk/parser.mly

Revision: 110
Author: maninder.sonu@gmail.com
Date: 3:17:10 PM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_array_ret_func.kaml
```

```
Revision: 111
Author: maninder.sonu@gmail.com
Date: 3:59:29 PM, Thursday, December 18, 2008
Message:
simple assignent bug solved. a=b  was not working.
----
Modified : /trunk/interpreter.ml

Revision: 112
Author: maninder.sonu@gmail.com
Date: 4:00:02 PM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_for1.kaml

Revision: 113
Author: maninder.sonu@gmail.com
Date: 4:13:54 PM, Thursday, December 18, 2008
Message:

----
Added : /trunk/tests/test_fib.kaml

Revision: 114
Author: kaorisu0611
Date: 4:40:38 PM, Thursday, December 18, 2008
Message:
* Bug xx+2 fixed
* escape seq "\\" for string added
----
Modified : /trunk/interpreter.ml
Modified : /trunk/scanner.mll

Revision: 115
Author: aankushgoel
Date: 6:30:34 PM, Thursday, December 18, 2008
Message:
calculates deteminant of n*n matrix
----
Added : /trunk/tests/determinant.kaml

Revision: 116
Author: aankushgoel
Date: 8:32:32 PM, Thursday, December 18, 2008
Message:
```

problem with array expression solved..now range and all are working again..

Modified : /trunk/interpreter.ml

Revision: 117
Author: maninder.sonu@gmail.com
Date: 9:25:57 PM, Thursday, December 18, 2008
Message:
For hash issue and the assignment of nth row or nth column of 2-D to 1-D and vice versa
----
Modified : /trunk/interpreter.ml

Revision: 118
Author: maninder.sonu@gmail.com
Date: 9:27:16 PM, Thursday, December 18, 2008
Message:

Added : /trunk/tests/test_array_transpose.kaml

Revision: 119
Author: aankushgoel
Date: 9:56:18 PM, Thursday, December 18, 2008
Message:
small change in interpreter..to let range work for 1D

Modified : /trunk/interpreter.ml

Revision: 120
Author: aankushgoel
Date: 12:23:05 AM, Friday, December 19, 2008
Message:
More test cases..

Added : /trunk/tests/arraySingleAssign1.kaml
Added : /trunk/tests/arraySingleAssign1.out
Added : /trunk/tests/arraySingleAssign2.kaml
Added : /trunk/tests/arraySingleAssign2.out
Added : /trunk/tests/differenceArraySingle3.kaml
Added : /trunk/tests/differenceArraySingle3.out
Added : /trunk/tests/differenceArraySingle4.kaml
Added : /trunk/tests/differenceArraySingle4.out
Added : /trunk/tests/intersectArraySingle3.kaml
Added : /trunk/tests/intersectArraySingle3.out
Added : /trunk/tests/intersectArraySingle4.kaml
Added : /trunk/tests/intersectArraySingle4.out

```
Added : /trunk/tests/minusSingle2.kaml
Added : /trunk/tests/minusSingle2.out
Added : /trunk/tests/unionArraySingle3.kaml
Added : /trunk/tests/unionArraySingle3.out
Added : /trunk/tests/unionArraySingle4.kaml
Added : /trunk/tests/unionArraySingle4.out

Revision: 121
Author: aankushgoel
Date: 12:49:19 AM, Friday, December 19, 2008
Message:
More test cases
----
Added : /trunk/tests/differenceArrayMulti3.kaml
Added : /trunk/tests/differenceArrayMulti3.out
Added : /trunk/tests/intersectArrayMulti3.kaml
Added : /trunk/tests/intersectArrayMulti3.out
Added : /trunk/tests/minusMulti2.kaml
Added : /trunk/tests/minusMulti2.out
Added : /trunk/tests/minusMulti3.kaml
Added : /trunk/tests/minusMulti3.out
Added : /trunk/tests/unionArrayMulti3.kaml
Added : /trunk/tests/unionArrayMulti3.out
Added : /trunk/tests/unionArrayMulti4.kaml
Added : /trunk/tests/unionArrayMulti4.out

Revision: 122
Author: maninder.sonu@gmail.com
Date: 12:51:33 AM, Friday, December 19, 2008
Message:
2-D array assignment do it [][2]b = [][]c
----
Modified : /trunk/interpreter.ml

Revision: 123
Author: aankushgoel
Date: 1:30:48 AM, Friday, December 19, 2008
Message:
finally two last test cases for the code that maninder just fixed.
----
Added : /trunk/tests/arrayMultiAssign1.kaml
Added : /trunk/tests/arrayMultiAssign1.out
Added : /trunk/tests/arrayMultiAssign2.kaml
Added : /trunk/tests/arrayMultiAssign2.out

Revision: 124
```

```
Author: aankushgoel
Date: 11:57:20 AM, Friday, December 19, 2008
Message:
implementing queue and stacks..it shows insert and delete features of out language..and
how the size of array is changes dynamically, plus it displays nesting of arithmetic
expression as an array index
----
Added : /trunk/tests/queueAndStack.kaml
Added : /trunk/tests/queueAndStack.out

Revision: 125
Author: aankushgoel
Date: 1:11:32 PM, Friday, December 19, 2008
Message:
these algorithms updated for output in proper alignment
----
Modified : /trunk/tests/determinant.kaml
Added : /trunk/tests/determinant.out
Modified : /trunk/tests/queueAndStack.kaml
Modified : /trunk/tests/queueAndStack.out

Revision: 126
Author: maninder.sonu@gmail.com
Date: 2:48:05 PM, Friday, December 19, 2008
Message:

----
Modified : /trunk/tests/test_array_transpose.kaml

Revision: 127
Author: maninder.sonu@gmail.com
Date: 2:50:55 PM, Friday, December 19, 2008
Message:

----
Added : /trunk/tests/test_array_transpose.out


Revision: 128
Author: aankushgoel
Date: 4:25:48 PM, Friday, December 19, 2008
Message:
inverse done...hurray
----
Added : /trunk/tests/inversetry.kaml
Added : /trunk/tests/inversetry.out
```

```
Revision: 129
Author: aankushgoel
Date: 5:38:28 PM, Friday, December 19, 2008
Message:
inverse try
----
Modified : /trunk/tests/inversetry.kaml

Revision: 130
Author: aankushgoel
Date: 5:40:36 PM, Friday, December 19, 2008
Message:
more test cases...all for exceptions..
----
Added : /trunk/tests/farrayMultiInitialize.kaml
Added : /trunk/tests/farrayMultiInitialize.out
Added : /trunk/tests/farrayMultiInitialize1.kaml
Added : /trunk/tests/farrayMultiInitialize1.out
Added : /trunk/tests/fdiffArray1.kaml
Added : /trunk/tests/fdiffArray1.out
Added : /trunk/tests/fdiffArrayMulti1.kaml
Added : /trunk/tests/fdiffArrayMulti1.out
Added : /trunk/tests/fintersectArray1.kaml
Added : /trunk/tests/fintersectArray1.out
Added : /trunk/tests/fintersectArrayMulti1.kaml
Added : /trunk/tests/fintersectArrayMulti1.out
Added : /trunk/tests/fintersectArrayMulti2.kaml
Added : /trunk/tests/fintersectArrayMulti2.out
Added : /trunk/tests/fminusSingle.kaml
Added : /trunk/tests/fminusSingle.out
Added : /trunk/tests/fminusSingle1.kaml
Added : /trunk/tests/fminusSingle1.out
Added : /trunk/tests/fsimpleArrayAssign.kaml
Added : /trunk/tests/fsimpleArrayAssign.out
Added : /trunk/tests/funionArray.kaml
Added : /trunk/tests/funionArray.out
Added : /trunk/tests/funionArray1.kaml
Added : /trunk/tests/funionArray1.out
Added : /trunk/tests/funionArrayMulti1.kaml
Added : /trunk/tests/funionArrayMulti1.out
Added : /trunk/tests/funionArrayMulti2.kaml
Added : /trunk/tests/funionArrayMulti2.out

Revision: 131
Author: maninder.sonu@gmail.com
Date: 5:44:26 PM, Friday, December 19, 2008
```

Message:

```
----
Added : /trunk/outfiles
Added : /trunk/outfiles/test_arith_div.out
Added : /trunk/outfiles/test_arith_sub.out
Added : /trunk/outfiles/test_arith_sum.out
Added : /trunk/outfiles/test_arith_sum_div.out
Added : /trunk/outfiles/test_array_ret_func.out
Added : /trunk/outfiles/test_array_transpose.out
Added : /trunk/outfiles/test_concat.out
Added : /trunk/outfiles/test_cond.out
Added : /trunk/outfiles/test_delete.out
Added : /trunk/outfiles/test_div_fail.out
Added : /trunk/outfiles/test_fib.out
Added : /trunk/outfiles/test_for1.kaml
Added : /trunk/outfiles/test_func.kaml
Added : /trunk/outfiles/test_func.out
Added : /trunk/outfiles/test_hash.out
Added : /trunk/outfiles/test_insert.out
Added : /trunk/outfiles/test_insert_fail.out
Added : /trunk/outfiles/test_mutual_functioncalls.out
Added : /trunk/outfiles/test_plus.out
Added : /trunk/outfiles/test_plus_fail.out
```