Lior Attias, Shida Jing, Annie Wang, Andrew Gorovoy, Bailey Nozomu Hwa
Xirtam Overview

- Xirtam is a matrix manipulation language with:
  - The functionality of Python’s **Numpy**
  - Syntax of **C** to maximize readability
  - Excel-like function calling
  - Unique features aimed at **saving users’ time**.
- Users: data scientists and frequent excel users (business analysts) who are:
  - Familiar with working with matrices
  - In need of something more powerful than Excel
  - Novice-to intermediate Programmers
- Uses:
  - For the said individuals to learn how to code
  - Quick and easy operations with matrices
4 Data Types

Num: Catch all for numbers, whether ints, floats, or doubles

Xirtam: 2-Dimensional Matrix of Numeric Values or Expressions

Bool: Basic Boolean type

String: Basic String type
Convenient Features

- Few Data Types
- Convenient Error checking
  - Checks for uninitialized variables
- Autocorrection to **Save Time**:
  - Matrices
    - Built-in Matrix functions autocorrect user input where possible
  - Functions
    - Automatic return values if a return value is not specified
  - For main()
    - LLVM expected int type
    - We added hidden int type to make the entry point an int in the back: as long as the user names the entry point main, it will work
num global1;
num global2;

num add(num a, num b) {
    num c;
    c = a + b;
    return c;
}

num main(){
    string s; bool b; xirtam m;
    s = "Hello";
    global1 = 1;
    global2 = 2;
    b = true;
    m = [[1, 2],[4, 5]];
    add(global1, global2);
    printn(global1); /* 1 */
    printm(m);
    /* 1.00 2.00
    4.00 5.00*/
}
Control Flow

```java
if (true && true) {
    println(1);
} else {
    println(0);
}
```

- **If ... else statement**

```java
num i;
for (i = 0 ; i < 5 ; i = i + 1) {
    println(i);
}
```

- **For loop**

```java
num i;
i = 5;
while (i > 0) {
    println(i);
    i = i - 1;
}
```

- **While loop**
Matrix Functionalities

- Easy initialization of a matrix
- User can easily get the following attributes of matrices:
  - Transpose of matrix
  - Rows
  - Columns
  - Number of rows
  - Number of columns
- User can easily perform the following transformations
  - Add two matrices
    - Component-wise addition
  - Subtract two matrices
  - Multiply two matrices
  - Get/Set specific index values

***All matrix methods have error checking***
Matrix Initialization

Initialization of Matrix

```c
num main() {
    xirtam m;
    m = [[1, 2], [3,4], [5,6]];
}
```

Creates 3 x 3 matrix with 1 in each index

Internal Logic

```c
matrix* initMatrix(double* listOfValues, int num_cols, int num_rows) {
    double* matrixValues = malloc(num_rows * num_cols * sizeof(double*));
    for(int r = 0; r < num_rows; r++) {
        for(int c = 0; c < num_cols; c++) {
            int idx = c + (r * num_cols);
            matrixValues[idx]=listOfValues[idx];
        }
    }
    //return a pointer to matrix struct
}
```
Matrix Attribute Examples

Number of Rows/Columns

```c
num main() {
    xirtam m;
    num result;

    m = [[[1,2,1,3],[1,2,3],[1,2,2,3]];
    result = getrows(m);

    printn(result);
}
```

### 3

Transpose

```c
num main() {
    xirtam m;

    m = [[1, 2], [3, 4]];
    printm(trans(m));
}
```

### 1.00  3.00
2.00 4.00
Matrix Function Examples

Add/Subtract

```c
num main() {
    xirtam m;
    m = [[1, 2], [3, 4], [5, 6]];
    printm(matadd(m,m));
}
```

```
###  2.00    4.00
6.00   8.00
10.00  12.00
```  

Multiplication

```c
num main() {
    xirtam m;
    xirtam n;
    xirtam ret;
    num r;
    m = [[4, 2,1], [422, 21], [0.4, 6.2]];
    n = [[1, 2, 3], [0.5, -1.2, 0]];
    ret = matmult(m, n);
}
```

```
Fatal error: exception Failure("No staggered Matrices allowed, rows must be same size")
```
Compiler Architecture with data flow
Future work

- Add in more complex built in matrix operations
- Increase the flexibility of matrix operations even more! Allow a user to concat matrices of different sizes, etc
- Image processing, tensorflow-- implement the basic functionalities to users to build complex programs
Demo

Problem: given a binary matrix $X$ and a zero matrix $Y$, build up information about clusterings of 1’s in $X$, and store the information in $Y$. 
Thank you for your help

Thank you, Professor Edwards and all the TA’s for all the help you provided.

Citation: this language is built upon MicroC and past project Matrx.