COMS W4115
Programming Languages and Translators

- FINAL PROJECT –

skyEM

(**S**eungjin **K** **Y**onghan **E**xam **M**aker)

Seungjin Nam sn2119@columbia.edu

Yonghan Kim (Group Leader) yk2081@columbia.edu

Table of Contents

Chapter 1

An Introduction to skyEM

1.1 Background

The increasing popularity of Internet and online education systems, from mere tutorials to an extensive university classes, demand some ways to effectively create and manage their testing systems. For example, TOEFL (Test of English as a Foreign Language) is now exclusively offered online as an iBT (Internet-based test) and many other computer related certification exams are conducted on computers.

Many programming languages already exist now such as C, C++, and JAVA, and programmers could use those languages to create and manage test sessions. However, it would take a lot of time and efforts for programmers just to create a small test, and therefore wasting valuable resources otherwise they could have used to focus on other things. To provide quick and easy way to create and implement testing schemes, skyEM language has been developed.

1.2 Language Description and Goals

skyEM is an interpretive language developed for the practical purpose of creating quizzes and tests. In this section, we state our goals for skyEM.

1.2.1 Easy-To-Use

skyEM has only few set of keywords and their associated parameters. This allows test makers to focus on the quality of questions and furthermore increasing the quality of educational experience as a whole.

1.2.2 Efficiency

Many of its built-in commands and functions will enable programmers to create user-friendly environment for the test takers without extensive typing. Also a number of valuable data such as test score and test percentage can be manipulated.

1.2.3 Portable

skyEM code will be compiled into Java byte code, which makes it architecturally neutral. It can be developed on any platform that has Java compiler, JVM and GUI. The code itself is also written in a plain ASCII text file, which enables it to be transferred to any other machine.

1.2.4 Object-Oriented Approach

User can create number of exams in one program, and each of the exams will have its own questions and properties. This is similar to creating objects in a conventional programming language.

1.3 Language Features

In this section, we describe some of the features of skyEM.

1.3.1 Pre-Defined Testing Format and Built-In-Functions

This will be the highlight of skyEM language. Pre-defined testing format allows users to supply just questions and answers and skyEM language will take care of the testing format. There will be a multiple-choice type questions, a matching type questions, true false type questions, and one-word-answer type questions. Built-in-functions will let the users to load mp3 files and jpg files, and do other miscellaneous tasks.

1.3.2 Advanced Testing Administrative Functions

Users will be able to set time limit for the exam. Also each question has its own value to it, meaning that more difficult problems will have higher point values. This can be useful if the user want to assign different point values for each question in the exam.

1.3.3 Data Types

skyEM will feature basic data types *integer, float, boolean,* and *String.* Each data type

has its own applicative operations. No type declaration is needed.

1.3.4 Flow Control

skyEM will support *if-else*, *while* and *break* statements. Since this is not intended for highly mathematical programs, other flow control statements like *for* is not included.

1.3.5 Internal Functions

The following necessary set of internal functions are included; *prnt*, *print*, *load*, and *return.*

Chapter 2
Language Tutorial

1. Getting Started

Before we actually analyze the sample codes, we need to make sure you have all the necessary components installed in your computer. J2SDK 1.4.2 or newer version is required to compile the source codes of skyEM and running them. Once you have compiled all the source codes, you can execute your program in the following manner.

```
OS$ java skyEM filename.em
```

Once the program is finished, it will return to the operating system.

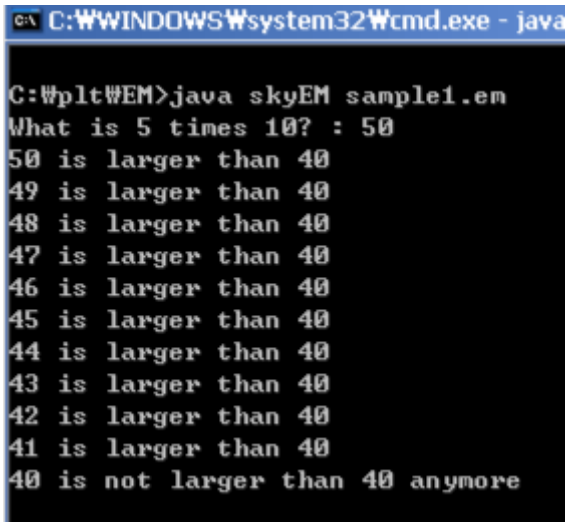
2. Simple Statements

A programming language is best taught by examples. Following is the source code for sample1.em program. Let us take a look at it.

```
/*
        Author  : Yonghan Kim
        File    : sample.em
*/
a = 5;
b = 10;
c = a * b;
temp = "What is 5 times 10? : ";

print(temp);
prtl(c);

while
{
        if ( c > 40 )
        {
                print(c);
                prtl(" is larger than 40");
                c = c - 1;
        }
        else
        {
                print(c);
                prtl(" is not larger than 40 anymore");
                break;
        }
}
```



The output sample1.em

First of all, anything from /* to */ is considered as comments and skyEM does not interpret them. // can be used for single line comments.

No declaration of variables are needed in skyEM. Just assign a *int, double, boolean,* or, *String* value to a variable and it is stored as above.

```
a = 5;
b = 10;
c = a * b;
temp = "What is 5 times 10? : ";
```

Next four lines of code initializes each variable a, b, c, and temp with integer 5, 10, 5 * 10, and a string “What is 5 times 10? :”.

```
print(temp);
prtl(c);
```

Also notice the difference between print(arg) and prtl(arg) functions. print(arg) will print the argument to the output only but prtl(arg) function also changes to the next line. Argument can be any expression.

Next whole block starting shows how to implement *while* and *if-else* statements. Since skyEM does not take condition in the *while* loop and therefore, it must have *break*

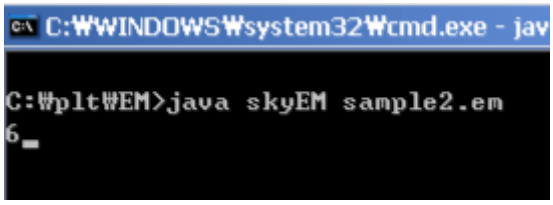
statement to control the loop. This is why we have *if-else* block inside the *while* loop. Notice that *if-else* block does take a conditional expression.

3. Using functions

Now let us learn how to define a function and use them. Take a look at sample2.em

```
answer = 3;
func add(x, y, z)
{
        answer = x + y + z;
        return answer;
}

a = foo(1,2,3);;
print(a);
```



output of sample2.em

To define a function, start with “func” and the name of the function, followed by arguments to be processed in the parenthesis. Now, any statements declared inside the {} will be executed when the function is called. To invoke the function, just type the function name and the arguments. Since this function has a return statement also, you can assign a variable to the return value. As expected, the value of *answer* variable is replaced from 3 to 6 after the function is called.

Warning : Notice that the user need to put two semicolons when assigning a return value of a function to a variable, one for calling a function and one for assigning.

4. Making Exams

Creating exam sessions is the heart of skyEM and the next sample will demonstrate

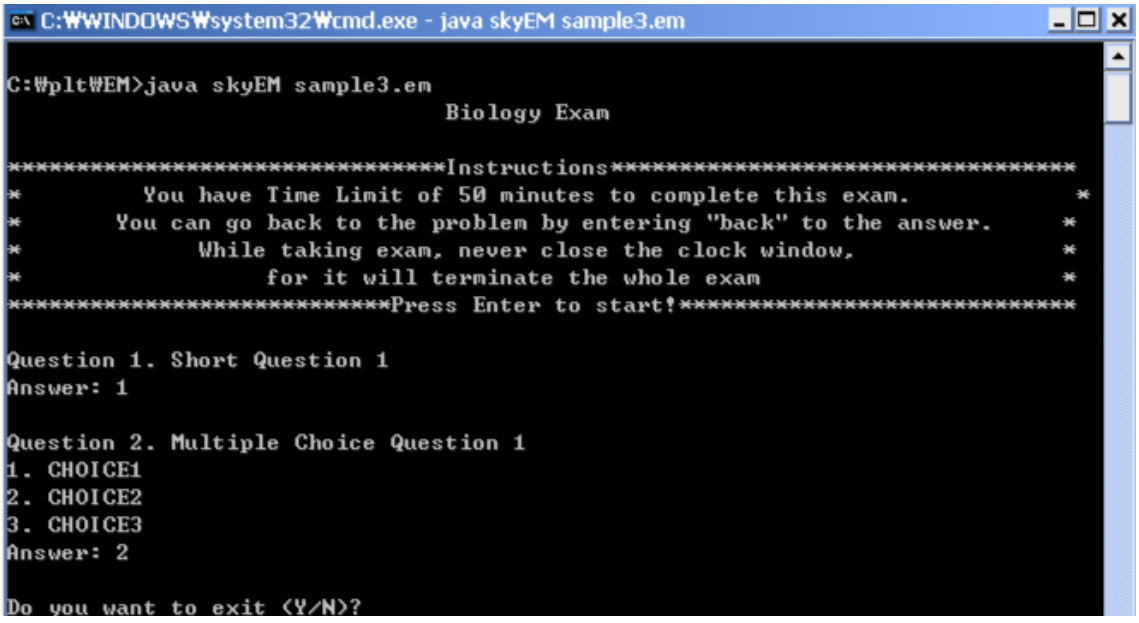
how to do it. Take a look at sample3.em.

```
SUBJECT.total(1);

subject Bio("Biology", 50)
{
        load.jpg("na.jpg");
        mult3 5# "Multiple Choice Question 1" : "CHOICE1", "CHOICE2", "CHOICE3" => 1;
        short 5# "Short Question 1" => "Short Answer 1";

}

Bio.start();
Bio.result();
```



output of sample3.em

`SUBJECT.total(1);`

This declares that there are total of 1 subject in this program. It must come before you actually define any subjects.

```
subject Bio("Biology", 50)
```

This is the beginning of the subject Bio definition. The first argument inside the parenthesis represents this subject and the second definition sets the time limit of this subject in minutes.

```
load.jpg("na.jpg");
mult3 5# "Multiple Choice Question 1" : "CHOICE1", "CHOICE2",
"CHOICE3" => 1;
```

load.jpg() function loads a jpg file whose filename is the argument. This function can only be called inside the subject body and it must come right before the question to be asked with the picture. mult3 represents a multiple choice questions type with 3 choices, and 5# indicates that its point value is 5. The following string is the question to be asked and the choices strings are written after the colon separated with comma. => represents answer in skyEM and therefore the first string in the choice is the answer to this question. There are two more multiple choice type questions, mult4 and mult5.

```
short 5# "Short Question 1" => "Short Answer 1";
```

This is another questions type, a short answer question. 5# again represents the point value of this program and the following string is the question. Of course as indicated with =>, the next string is the answer to this program. There is also true-false and match type questions. Please refer to the language reference manual for instruction.

```
Bio.start();
```

This function will start the exam session for Biology subject.

```
Bio.result();
```

This function will display the result to the screen.

Chapter 3

Language Reference Manual

1. Lexical Conventions

1.1 Comments

Two types of comments are used in skyEM.

multi-line comments : starts with characters ‘ /* ’ and terminated by ‘ */ ’.
single line comments : starts with characters ‘ // ’ and continues until the end of the line.

1.2 Identifiers

An identifier consists of digits, letters and underscore ‘ _ ’, and it is case sensitive. The first character must be a letter or underscore.

1.3 Keywords

These words are reserved, therefore cannot be used as identifiers.

if	else	break	subject	load
short	mult3	mult4	mult5	truce
false	match	print	prtl	while

1.4 Numbers

Numbers consist of digits and for floating point numbers, optional decimal point ‘.’ can be used. Integer numbers and floating point numbers will be distinguished.

1.5 String Literals

String literals are anything enclosed by a set of double quotes “ “. To have double quotes inside the string, put double quotes twice, “”.

1.6 Other tokens

There are a few reserved characters or pairs of characters that must be used correctly in the language. The reserved characters and pairs are as follows.

```
{       }       (       )       [       ]       =>
+       ++      -       --      *       /       %
=       = =     <=      >=      <       >       !=
#
```

2. Types

`boolean`	:	Boolean values that can either be true or false
`int`	:	standard 32-bit integers
`double`	:	64-bit IEEE floating format
`string`	:	a string of characters

3. Expression

3.1 Primary Expression

Primary expressions have identifiers, constants, strings, or expressions in parentheses.

Primary-expression: *identifier*
| *string*
| *(expression)*

A parenthesized expression is a primary expression, which returns the value of enclosed expression. The presence of parentheses makes the precedence higher.

3.2 Arithmetic Expressions

Arithmetic Expressions take primary expressions as operands.

Multiplicative Operators

The binary operators ‘ * ‘, ‘ / ‘, ‘ % ‘ indicate multiplication, division, and modulo respectively. They are grouped left-to-right. They are only applicable to int, double, float.

Additive Operators

the binary operators ‘ + ‘, ‘ – ‘ indicate addition and subtraction, respectively. They are grouped from left to right. They are applicable to int, double, float.

3.3 Relational Expressions

Binary relational operators ‘ >= ’, ‘ <= ’, ‘ = = ’, ‘ != ’, ‘ > ’ and ‘ < ’ indicate whether the first operand is greater than or equal to, less than or equal to, equal to, not equal to, greater than, or less than the second operand, respectively.

3.4 Logical Expression

Logical operators take relational expressions as operands. Or operator ‘ || ’ indicates the logical or of two relational; expressions. It has the lowest precedence. And operator ‘ && ’ indicates the logical and of two relational expressions. It has higher precedence than or operator.

4. Statements

Statements are basic elements of the program. A sequence of statements are executed sequentially, unless flow-control statements indicate otherwise.

4.1 Statements in ‘ { ‘ and ‘ } ’

If a group of zero or more statements can be surrounded by ‘ { ‘ and ‘ } ’, then they are grouped and run together in a sequential manner.

4.2 Assignments

To assign right valued expression to left valued expression,

left-valued expression = right-valued expression;

4.3 Conditional Statements

Conditional statements have two forms. The first form is,

```
if ( relational expression )
{
        statements;     }
```

The statement is executed if the relational expression is evaluated to be true. Otherwise, it skips the statements within the curly brackets and continues the program. The second form is,

```
if ( relational expression )
{
        statements;
}
else
{
        statements;
}
```

This form works same as the first form, but the relational expression is evaluated to be false, it executes the statements enclosed by else curly bracket.

4.4 Loops

All loops in the language take the form of a while loop. To use the while loop,

```
while
{
        statements;
        if ( relational expression )
                break;
}
```

Unlike other traditional programs, skyEM does not take condition for *while* loop. Instead it has a *if* statement which contains *break;*. Break statement is introduced to break out from the loop and continue the program. To use break statements, type keyword break followed by a statement terminator ‘ ; ‘

5. General Function Calls

Function call takes the following form,

function_name(parameter list);

Function must be defined before it is called. Parameter list must take the same type of variables as defined previously.

5.1 Return statements

Return statement is used within the function definition body to return the specific value. A basic return statement would look like,

return return_value;

5.2 Function Definition

To define a function,

```
func function_name ( parameter list )
{
   statements;
   return return_type;
}
```

Function_name is the name of the function that user has given and the parameter list is a list of identifier of arguments, separated by commas ‘ , ’.
Parameter list can be empty.

5.3 Console output function

The function *print()* has two different versions. The first version takes a variable as an argument and prints out the value of the variable. It takes the form,

print(variable);

The second version takes a string and prints it out to the screen. The string must

be enclosed in double quotes and takes the form,

print(“This will be printed.”);

This is also true for *prtl()* function, which will print the argument and put a newline character at the end.

5.4 File I/O Functions

Function load() will load a picture and display it or load a sound file and play it. Image file can be jpg and sound file can be mp3. This function can be only used in the subject body, right before the question to be asked. To use *load()* function,

load(file_type, filename);

6. skyEM Special Features

6.1 Subject Object

This object represents one examination with all of its questions and answers along with other properties, such as time limit, order of the questions, and the types of questions. It is declared in the following way

```
subject subject_name(String subject_name, int time )
{
        statements;
}
```

subject_name will be used to call the subject functions. The first argument is the actual name of the subject, which is the string value. The second integer argument represents the time limit for this particular examination.

6.2 Question Object

This object represents an individual question. It is created in the body of subject objects. There are several types of questions; multiple choice question, matching, true/false, one word answer and they are declared in the following manner;

```
mult3 5# "What is your first name?" : "Yonghan",
"Seungjin", "Stephen" => 1;

short 5# "What is your last name?" => "Edwards";

truce 5# "Are you a teacher?";

false 5# "Are you a student?";

match 5# “Question” : "MATCH1", "MATCH2", "MATCH3",
"MATCH4" => "ANSWER1", "ANSWER2", "ANSWER3", "ANSWER4";
```

6.3 Built in functions

There are several built-in scoring functions associated with subject object. First function is *total()* function, which returns total score of the examination. It is used as following,

subject_name.total();

The second function is *score()* function, which returns the exam score for the subject. It is used as following,

subject_name.score();

The fourth function is *start()* function, which starts the examination for the subject. It is used as following,

subject_name.start();

Chapter 4
Project Plan

1. Team Responsibilities

Two of our members decided not to take this class after the midterm, so we had to distribute the workloads slightly heavier to each individuals. Since there are only two individuals in the group, we did not really have regular meetings. However, we did set a goal for each member so we have some kind of division of labor. Here are the fundamental tasks for each member.

Yonghan Kim : Front-end, Treewalker, Architecture, Documentation

- Designed the basic language syntax and its architecture
- Created Lexer, Parser and Treewalker with antlr
- Contributed to the documentation

Seungjin Nam : Back-end, All Support Libraries, Testing, Documentation

- Created data types : int, double, string, boolean
- Created subject library including question types and subject class
- Conducted tests on the program
- Conducted to the documentation

2. Project Timeline

Here are the intended deadlines for each specific tasks.

9 / 27	: Proposal due in class
Second week of October	: Language grammar set
10 / 20	: Language Reference Manual due in class
Third week of November	: Front-End
First week of December	: Back-End
Second Week of December	: Testing, Documentation
12 / 20	: Final Report due

3. Software Development Environment

3.1 Java 1.5

With the exception of lexer, parser and the treewalker, all other files were written in Java. Because of this, this program can be run on any system that has Java VM running. We liked Java since it has very clear coding style and there are many available supporting libraries

3.2 Antlr

Without Antlr, there was no way we could develop such a language. At first, it seemed impossible to learn but as time went on, we realized how it made simple to create language grammar. We used Antlr to develop lexer, paresr and the treewalker.

3.3 EditPlus 2

This is a free text editing tool that helped us tremendously. It has automatic indentation and it changes the color of the text according to the type. For example, keywords are blue and the objects are red for Java codes.

4. Project Log

We have listed the dates for the significant project developments.

9 / 20	: First team meeting, decided to what to do
9 / 27	: Proposal paper finished
10 / 5	: One member stopped taking this class
10 / 16	: First version of grammar
10 / 20	: Language Reference Manual finished
11 / 7	: First version of EMgrammar.g
11 / 20	: Data type classes finished
11 / 29	: First version of Subject and questions types
12 / 12	: Workable version of treewalker
12 / 15	: Began testing
12 / 18	: Began documentation
12 / 19	: Final version of final report

5. Lessons Learned

Yong Han Kim: I realized that developing a compiler or a translator is much different from developing a program. It was really difficult, especially generating grammars and creating tree walker. However, I am glad now because I learned so much since I started working on the project. The first lesson I learned is an obvious one. Start early. Now I feel like I could do so much more if I had more time. Also trust your teammates because nothing can work in a group when there is no trust. Finally, I urge anyone who is first trying to create a compiler, to look at how others did. It will greatly help you get started.

SeungJin Nam: Doing this project I have learned a lot of things about program and especially about java. I realized how much I did not know about the java programming, and other general programming structures. I got to learn some of GUI programming, swing also, and it was very different from what I have learned until now. Multi threading programming was difficult, but it was very fun to see how the two things can run at the same time. But controlling those were very difficult task. Although I didn't do much about antlr, but it was also very rewarding for me to experience those tools to generate the compiler. This was my first time doing all these stuff, and it was very new for me.

Chapter 5

Architecture Design

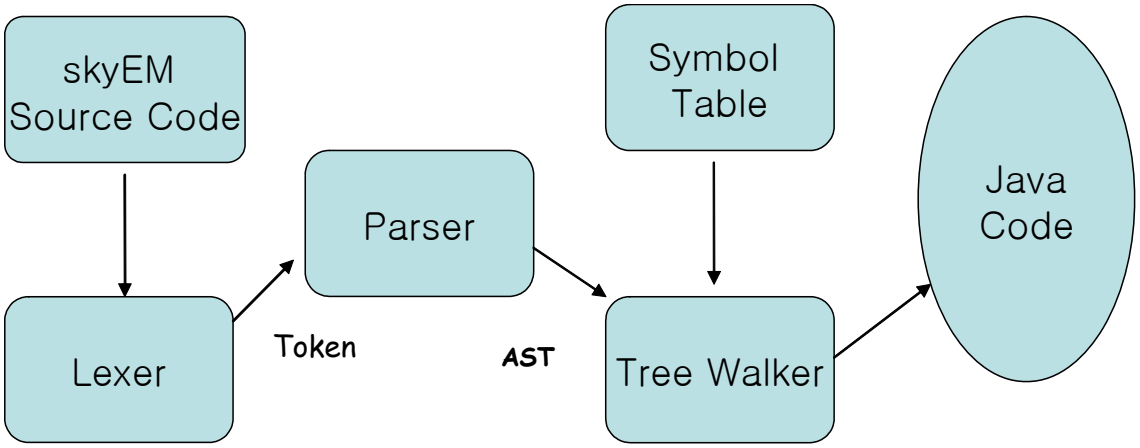


Figure 1. Streamline of architecture

There are two stages in front end another two stages in back end to build the program. So there are total of four main stages in the architecture of skyEM program. Each of they are interconnected by each other, passing on necessary information to the other. Therefore it cannot operate with one another.

First, important stage is the lexical analysis. The source code will go through the lexer generated by antlr, and the code will be stripped out to tokens. Then the second stage comes in, and those tokens are passed on to the parser of skyEM, to become Abstract Syntax Tree (AST) Structures. Each AST has their roots as tokens, and expand the tree with matched rules. Going through parser will check both syntax and semantic errors and return failure if it fails.

We designed skyEM to be an interpreter. Since skyEM is an interpreter program, it does not go through code generation; it goes through the tree walker straight away. In tree walker stage, previously made AST will be applied one by one to the tree walker rules generating symbol table which will convert the skyEM source doe to java code. Therefore, in each step, it matches with the java source code that we have written (classes, functions, variables etc) for skyEM to generate the corresponding java codes. And all the four stages have been done, and the program will run if no errors were found.

Chapter 6
Test Plan

1. Goal

Software testing is a process of operating the program for the purpose of finding bugs. It would be impossible to catch all bugs in a problem because testing of every possible input to the program is just not possible. However, with carefully trying possible cases as the development goes on, it is possible to reduce the amount of bugs after the final product has been created.

2. Method

Each team member was responsible for testing the code separately in isolation so it does not affect others in the development. The following testing methods were used to find and analyze the bugs. In all tests, looking at the AST from the parser was incredibly helpful. We did not have an automated procedure to find the bugs.

2.1 Module Testing

Whenever a new function was made, it has to be tested thoroughly before it can interact with other functions in the program. This was done by writing a sample program using only the function that was newly created. It must be tested with enough cases or the testing is meaningless.

2.2 Regression Testing

Once a module or a function passed the module testing, then the regression testing should be implemented next. This step is important because it ensures that in the course of making the required changes to the product, no other unintended changes were made. This can be done by comparing the output with the previous test cases.

3.3 Sample Testing

Here is a sample code from test8.em.

```
/*
	TEST CASE 8
	Author Yonghan Kim

*/

SUBJECT.total(1);		// total number of subjects must be declared at first line

subject Bio("Biology", 50)
{
	load.jpg("na.jpg");
	mult3 5# "Multiple Choice Question 1" : "CHOICE1", "CHOICE2", "CHOICE3" => 1;
	mult4 10# "Multiple Choice Question 2" : "CHOICE1", "CHOICE2", "CHOICE3", "CHOICE4" => 4;
	short 5# "Short Question 1" => "Short Answer 1";
	truce 5# "True Question 1";
	false 5# "False Question 1";

}



Bio.start();		// starting test

a = 3;

func foo(x, y, z)		// testing function declaration
{
	a = x + y + z;
	return a;
}

a = foo(1,2,3);;		// testing function call
print(a);
```

```
if ( a == 6 )                // testing if
{
        Bio.result();
        prtl("""GOOD""");    // testing double quotes in a string
}
else
        prtl("""BAD""");

prtl("");                    // printing empty line
print("a");                  // difference between print and prtl


i = 0;
while
{
        i = i + 1;
        Math.result();
        if ( i == 6 )        // Math score will print out 6 times
                break;
}
```

This is the output of the program when executed.

```
                              Biology Exam

*******************************Instructions*********************************
*        You have Time Limit of 50 minutes to complete this exam.           *
*      You can go back to the problem by entering "back" to the answer.     *
*           While taking exam, never close the clock window,              *
*                for it will terminate the whole exam                    *
***************************Press Enter to start!*****************************

copy is  Thread[main,5,main]
Question 1. Multiple Choice Question 2
```

1. CHOICE1

2. CHOICE2

3. CHOICE3

4. CHOICE4

Answer: 1

Question 2. Multiple Choice Question 1

1. CHOICE1

2. CHOICE2

3. CHOICE3

Answer: 2

10

**********Score Report**********

1. x 2. x

****

Your Score is 0.0!

"GOOD"

aHELLO

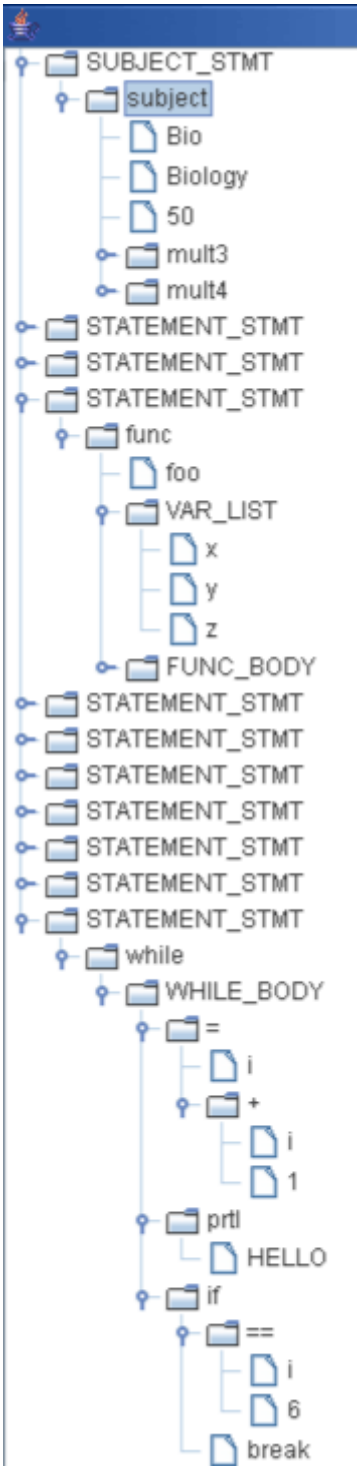
HELLO

HELLO

HELLO

HELLO

HELLO



the output of tree

Chapter 7
Lesson Learned

```
/*
        Grammar file for skyEM
        Author : Yonghan Kim
*/

class skyEMLexer extends Lexer;

options {
    k = 3;
    charVocabulary = '\3'..'\377';
    testLiterals = false;
    exportVocab = skyEM;
}

protected
LETTER   : 'a'..'z' | 'A'..'Z' | '_' ;

protected
DIGIT   :   '0'..'9';

WS      : (' ' | '\t' | '\n' {newline(); } | '\r')
                { $setType(Token.SKIP); };


COMMENT : ( "/*" (
                    options {greedy=false;} :
                    ( '\n' | '\r' )     {newline();}
                    | ~( '\n' | '\r' )
                )* "*/"
            | "//" (~( '\n' | '\r' ))* {newline();}
          )                         { $setType(Token.SKIP); }
        ;

COLON   : ':';
DOT     : '.';
LOAD    : "<<";
ANSWER  : "=>";
LPAREN  : '(';
RPAREN  : ')';
MULT    : '*';
PLUS    : '+';
MINUS   : '-';
DIV     : '/';
MOD     : '%';
SEMI    : ';';
LBRACE  : '{';
RBRACE  : '}';
LBRK    : '[';
RBRK    : ']';
ASSIGN  : '=';
COMMA   : ',';
GE      : ">=";
LE      : "<=";
GT      : '>';
LT      : '<';
EQ      : "==";
NEQ     : "!=";
NOT     : '!';
POINT   : '#';


ID      options { testLiterals = true; }
        : LETTER (LETTER | DIGIT)*
        ;


NUMBER  : (DIGIT)+
          ('.' (DIGIT)*)?
          (('E'|'e') ('+'|'-')? (DIGIT)+)? ;


STRING  : '"'!
          ( ~('"' | '\n')
          |  ( '"'! '"')                // cannot make it to '\'
          )*
          '"'!  ;
```

```
 77
 78
 79
 80
 81  class skyEMParser extends Parser;
 82
 83  options{
 84      k = 2;
 85      buildAST = true;
 86      exportVocab = skyEM;
 87  }
 88
 89  tokens
 90  {
 91          PROGRAM;
 92          STATEMENT;
 93          VAR_LIST;
 94          EXPR_LIST;
 95          FUNC_CALL;
 96          ARRAY;
 97          WHILE;
 98          SUBJECT;
 99          SUB_BODY;
100          SUB_CALL;
101          STATEMENT_STMT;
102          SUBJECT_STMT;
103          SUBJECT_CALL;
104          START;
105          SCORE;
106          RETURN_SCORE;
107  }
108
109
110  {
111      int nr_error = 0;
112      public void reportError( String s ) {
113          super.reportError( s );
114          nr_error++;
115      }
116      public void reportError( RecognitionException e ) {
117          super.reportError( e );
118          nr_error++;
119      }
120  }
121
122  program
123          : (subject_stmt | statement_stmt )* EOF!
124                  { #program = #([PROGRAM,"PROGRAM"], program); }
125          ;
126
127  subject_stmt
128          : (subject | subject_dec )
129                  { #subject_stmt = #([SUBJECT_STMT, "SUBJECT_STMT"], subject_stmt); }
130          ;
131
132  statement_stmt
133          :  (statement | func_def)
134              {#statement_stmt = #([STATEMENT_STMT, "STATEMENT_STMT"], statement_stmt); }
135          ;
136
137
138  statement
139          : if_stmt
140          | while_stmt
141          | return_stmt
142          | attach_stmt
143          | assignment
144          | break_stmt
145          | print_stmt
146          | println_stmt
147          | func_call
148          | subject_call
149          | LBRACE! (statement)* RBRACE!
150              {#statement = #([STATEMENT,"STATEMENT"], statement); }
151          ;
152
```

```
153 subject_call
154         : ID DOT! sub_func
155                 {#subject_call = #([SUBJECT_CALL,"SUBJECT_CALL"], subject_call); }
156         ;
157
158 sub_func
159         : "start" LPAREN! RPAREN! SEMI!
160                 {#sub_func = #([START,"START"], sub_func); }
161         | "result" LPAREN! RPAREN! SEMI!
162                 {#sub_func = #([SCORE,"SCORE"], sub_func); }
163         | "score" LPAREN! RPAREN! SEMI!
164                 {#sub_func = #([RETURN_SCORE,"RETURN_SCORE"], sub_func);}
165         ;
166
167 subject_dec
168         : "SUBJECT"^ DOT! "total"! LPAREN! expression RPAREN! SEMI!
169         ;
170
171 subject : "subject"^ ID LPAREN! STRING COMMA! expression RPAREN! subject_body
172         ;
173
174 subject_body
175         : LBRACE! (question)* RBRACE!
176         ;
177
178
179 question
180         : "short"^ expression POINT! expression ANSWER! expression SEMI!
181         | "truce"^ expression POINT! expression SEMI!
182         | "false"^ expression POINT! expression SEMI!
183         | "mult3"^ expression POINT! expression COLON! expression COMMA! expression COMMA! e
184         | "mult4"^ expression POINT! expression COLON! expression COMMA! expression COMMA! e
185         | "mult5"^ expression POINT! expression COLON! expression COMMA! expression COMMA! e
186         | "match"^ expression POINT! expression (COMMA! expression)* ANSWER expression (COMM
187         | "load"^ DOT! ( "jpg"! | "mp3"! ) LPAREN! expression RPAREN! SEMI!
188         ;
189
190
191 if_stmt
192         : "if"^ LPAREN! expression RPAREN! statement
193           (options {greedy = true;}: "else"! statement )?
194         ;
195
196 while_stmt
197         : "while"^ while_body
198         ;
199
200 while_body
201         : LBRACE! (statement)* RBRACE!
202           {#while_body = #([STATEMENT,"WHILE_BODY"], while_body); }
203         ;
204
205
206 break_stmt
207         : "break"^ SEMI!
208         ;
209
210 return_stmt
211         : "return"^ (expression)? SEMI!
212         ;
213
214 attach_stmt
215         : "attach"^ STRING SEMI!
216         ;
217
218 print_stmt
219         : "print"^ LPAREN! ( expression ) RPAREN! SEMI!
220         ;
221
222 println_stmt
223         : "prtl"^ LPAREN! ( expression ) RPAREN! SEMI!
224         ;
225
226 assignment
227         : l_value ( ASSIGN^ ) expression SEMI!
228         ;
```

```
229
230 func_call
231         : ID LPAREN! expr_list RPAREN! SEMI!
232             {#func_call = #([FUNC_CALL,"FUNC_CALL"], func_call); }
233         ;
234
235 expr_list
236         : expression ( COMMA! expression )*
237             {#expr_list = #([EXPR_LIST,"EXPR_LIST"], expr_list); }
238         |   {#expr_list = #([EXPR_LIST,"EXPR_LIST"], expr_list); }
239         ;
240
241 func_def
242         : "func"^ ID LPAREN! var_list RPAREN! func_body
243         ;
244
245 var_list
246         : ID ( COMMA! ID )*
247             {#var_list = #([VAR_LIST,"VAR_LIST"], var_list); }
248         |
249             {#var_list = #([VAR_LIST,"VAR_LIST"], var_list); }
250         ;
251
252 func_body
253         : LBRACE! (statement)* RBRACE!
254             {#func_body = #([STATEMENT,"FUNC_BODY"], func_body); }
255         ;
256
257 expression
258         : logic_term ( "||"^ logic_term )*
259         ;
260
261 logic_term
262         : logic_factor ( "&&"^ logic_factor )*
263         ;
264
265 logic_factor
266         : (NOT^)? relat_expr
267         ;
268
269 relat_expr
270         : arith_expr ( (GE^ | LE^ | GT^ | LT^ | EQ^ | NEQ^) arith_expr )?
271         ;
272
273 arith_expr
274         : arith_term ( (PLUS^ | MINUS^) arith_term )*
275         ;
276
277 arith_term
278         : arith_factor ( (MULT^ | DIV^ | MOD^) arith_factor )*
279         ;
280
281 arith_factor
282         : r_value
283         ;
284
285
286 r_value
287         : l_value
288         | func_call
289         | subject_call
290         | NUMBER
291         | STRING
292         | "true"
293         | "false"
294         | LPAREN! expression RPAREN!
295         ;
296
297 l_value
298         : ID
299         ;
300
301 cl_statement
302         : ( statement | func_def )
303         | "exit"
304             { System.exit(0); }
```

```
305         | EOF!
306             { System.exit(0); }
307         ;
```

```
/*
        Tree walker file for skyEM
        Author : Yonghan Kim
*/



{
import java.io.*;
import java.util.*;
}

class skyEMWalker extends TreeParser;
options{
    importVocab = skyEM;

}

{
    static skyEMType null_data = new skyEMType( "<NULL>" );
    skyEMInterpreter ipt = new skyEMInterpreter();
    int counter = 0;
    int counter2 = 0;
    int index = 0;
    Subjects[] sub;
    String identifier = null;
    boolean go = true;
    boolean load = false;
    boolean match_questions = true;
    String data = null;

        private int get_index(String s)
        {
                int val = 0;
                int i;
                for( i = 0; i < sub.length-1; i++)
                {
                        if ( (sub[i].getID()).compareTo( s ) == 0 )
                        {
                                go = true;
                                break;
                        }
                        else
                        {
                                go = false;
                        }
                        val++;
                }
                return val;
        }

}


program
        :
                #(PROGRAM (body:. { subject(#body);} )* )
        ;

subject returns [ skyEMType r ]
{
        skyEMType a, b, c;
        skyEMType[] x;
        r = null;
}
        :

        | #(SUBJECT_STMT (subj:. { if ( ipt.canProceed() ) subject(#subj); } )*)
        | #(STATEMENT_STMT (stmt:. { if ( ipt.canProceed() ) expr(#stmt); } )*)

        | #("SUBJECT" a=expr)           // TOTAL NUMBER OF SUBJECT DECLARATION
                {       sub = new Subjects[a.to_int()+1]; }

        | #("subject" sname:ID a=expr b=expr    // SUBJECT DECLARATION
                {
                        identifier = sname.getText();
```

```
 77                         String name = a.toString();
 78                         int time = b.to_int();
 79                         sub[counter] = new Subjects(identifier, name, time);
 80                         counter++;
 81                 }
 82                         (body:. { if ( ipt.canProceed() ) subj(#body); } )* )
 83
 84         ;
 85
 86 subj returns [ skyEMType r ]
 87 {
 88         skyEMType a, b, c, d, e, f, g, h;
 89         r = null;
 90
 91 }
 92         : #("short" a=expr b=expr c=expr )
 93                 {       counter2 = counter - 1;
 94                         if ( !load )
 95                         sub[counter2].ShortAnswers( a.to_int(), b.toString(), c.toString() )
 96                         else
 97                         {
 98                                 sub[counter2].ShortAnswers( data, a.to_int(), b.toString(),
 99                                 load = false;
100                         }
101                 }
102         | #("truce" a=expr b=expr )
103                 {       counter2 = counter - 1;
104                         if ( !load )
105                         sub[counter2].ShortAnswers( a.to_int(), b.toString(), "true" );
106                         else
107                         {
108                                 sub[counter2].ShortAnswers( data, a.to_int(), b.toString(),
109                                 load = false;
110                         }
111                 }
112         | #("false" a=expr b=expr )
113                 {       counter2 = counter - 1;
114                         if ( !load )
115                         sub[counter2].ShortAnswers( a.to_int(), b.toString(), "false" );
116                         else
117                         {
118                                 sub[counter2].ShortAnswers( data, a.to_int(), b.toString(),
119                                 load = false;
120                         }
121                 }
122         | #("mult3" a=expr b=expr c=expr d=expr e=expr f=expr)
123                 {       counter2 = counter - 1;
124                         if ( !load )
125                         sub[counter2].MultipleChoice( a.to_int(), b.toString(), f.to_int(),
126                         else
127                         {
128                                 sub[counter2].MultipleChoice( data, a.to_int(), b.toString()
129                                 load = false;
130                         }
131                 }
132         | #("mult4" a=expr b=expr c=expr d=expr e=expr f=expr g=expr)
133                 {       counter2 = counter - 1;
134                         if ( !load )
135                         sub[counter2].MultipleChoice( a.to_int(), b.toString(), g.to_int(),
136                         else
137                         {
138                                 sub[counter2].MultipleChoice( data, a.to_int(), b.toString()
139                                 load = false;
140                         }
141                 }
142         | #("mult5" a=expr b=expr c=expr d=expr e=expr f=expr g=expr h=expr)
143                 {       counter2 = counter - 1;
144                         if ( !load )
145                         sub[counter2].MultipleChoice( a.to_int(), b.toString(), h.to_int(),
146                         else
147                         {
148                                 sub[counter2].MultipleChoice( data, a.to_int(), b.toString()
149                                 load = false;
150                         }
151                 }
152         | #("match" a=expr matchbody:.)
```

```
153                 {
154                         counter2 = counter - 1;
155                         sub[counter2].Matching( a.to_int() );
156                         match_support(#matchbody);
157                 }
158         | #("load" a=expr )     { data = a.toString(); load = true; }
159         | START   { if (go) sub[index].start("r"); else System.out.println("SUBJECT NOT DECL
160         | SCORE   { if (go) sub[index].result(); else System.out.println("SUBJECT NOT DECLAR
161         | RETURN_SCORE
162                 {
163
164                         if (go)
165                         {
166                                 double db = sub[index].score();
167                                 r = new skyEMDouble(db);
168                         }
169                         else
170                                 System.out.println("SUBJECT NOT DECLARED"); }
171         ;
172
173 match_support
174         : str:STRING
175                 {
176                         if ( match_questions )
177                         {
178                                 sub[counter2].addLeftSideMatching( str.getText() );
179                         }
180                         else
181                         {
182                                 sub[counter2].addRightSideMatching( str.getText() );
183                         }
184                 }
185                 matchbody2:.    { match_support(#matchbody2); }
186         | ANSWER        matchbody3:.
187                         {       match_questions = false;
188                                 match_support(#matchbody3);
189                         }
190         | SEMI          {       match_questions = true;
191                                 sub[counter2].MatchingSet();
192                         }
193         ;
194
195
196 expr returns [ skyEMType r ]
197 {
198
199     skyEMType a, b, c;
200     skyEMType[] x;
201     String s = null;
202     String[] sx;
203     r = null_data;
204     Vector v;
205 }
206
207
208         :
209         #(SUBJECT_CALL  id2:ID body2:.
210                 {       identifier = id2.getText();
211                         index = get_index(identifier);
212                 if ( ipt.canProceed() ) r = subj(#body2); } )
213
214         | #("||" a=expr right_or:.)
215             {
216                 if ( a instanceof skyEMBool )
217                     r = ( ((skyEMBool)a).var ? a : expr(#right_or) );
218                 else
219                     r = a.or( expr(#right_or) );
220             }
221         | #("&&" a=expr right_and:.)
222             {
223                 if ( a instanceof skyEMBool )
224                     r = ( ((skyEMBool)a).var ? expr(#right_and) : a );
225                 else
226                     r = a.and( expr(#right_and) );
227             }
228         | #(NOT a=expr)             { r = a.not(); }
```

```
229         | #(GE a=expr b=expr)        { r = a.ge( b ); }
230         | #(LE a=expr b=expr)        { r = a.le( b ); }
231         | #(GT a=expr b=expr)        { r = a.gt( b ); }
232         | #(LT a=expr b=expr)        { r = a.lt( b ); }
233         | #(EQ a=expr b=expr)        { r = a.eq( b ); }
234         | #(NEQ a=expr b=expr)       { r = a.ne( b ); }
235         | #(PLUS a=expr b=expr)      { r = a.plus( b ); }
236         | #(MINUS a=expr b=expr)     { r = a.minus( b ); }
237         | #(MULT a=expr b=expr)      { r = a.mult( b ); }
238         | #(DIV a=expr b=expr)       { r = a.div( b ); }
239         | #(MOD a=expr b=expr)       { r = a.mod( b ); }
240         | #(ASSIGN a=expr b=expr)      { r = ipt.assign( a, b ); }
241
242
243
244         | #(FUNC_CALL a=expr x=mexpr)
245            { r = ipt.funcInvoke( this, a, x ); }
246         | num:NUMBER                 { r = ipt.getNumber( num.getText() ); }
247         | str:STRING                 { r = new skyEMString( str.getText() ); }
248         | "true"                     { r = new skyEMBool( true ); }
249         | "false"                    { r = new skyEMBool( false ); }
250         | #("print" a=expr)             { a.print(); }
251         | #("prtl" a=expr)              { a.println(); }
252         | #("if" a=expr thenp:. (elsep:.)?)
253             {
254                 if ( !( a instanceof skyEMBool ) )
255                     return a.error( "if: expression should be bool" );
256                 if ( ((skyEMBool)a).var )
257                     r = expr( #thenp );
258                 else if ( null != elsep )
259                     r = expr( #elsep );
260             }
261         | #(STATEMENT (stmt:. { if ( ipt.canProceed() ) r = expr(#stmt); } )*)
262         | #("while" loopbody:.)
263                 {
264                         while ( ipt.canProceed() )
265                         {
266                                 r = expr( #loopbody );
267                                 ipt.loopNext( s );
268                         }
269                         ipt.loopEnd( s );
270                 }
271         | "break"                    { ipt.setBreak( s ); }
272         | #("return" ( a=expr        { r = ipt.rvalue( a ); }
273                      )?
274            )                         { ipt.setReturn( null ); }
275         | id:ID                   { r = ipt.getVariable( id.getText() ); }
276         | #("func" fname:ID sx=vlist fbody:.)
277             { ipt.funcRegister( fname.getText(), sx, #fbody ); }
278
279
280
281
282 //      | #("attach" a=expr)        { ipt.attach( a ); }
283 //      | #(ARRAY                 { v = new Vector(); }
284 //              ( a = expr       { v.add(a); }   ) )
285 //              { System.out.println("ARRAY"); }
286 //              ;
287 //      | #(
288
289
290
291 mexpr returns [ skyEMType[] rv ]
292 {
293     skyEMType a;
294     rv = null;
295     Vector v;
296 }
297         : #(EXPR_LIST          { v = new Vector(); }
298                 ( a=expr       { v.add( a ); }
299                 )*
300            )                   { rv = ipt.convertExprList( v ); }
301         | a=expr               { rv = new skyEMType[1]; rv[0] = a; }
302         |  #(FOR_CON           { v = new Vector(); }
303                 ( s:ID a=expr { a.setName( s.getText() ); v.add(a); }
304                 )+
```

```
305             )                { rv = ipt.convertExprList( v ); }
306         ;
307
308 vlist returns [ String[] sv ]
309 {
310     Vector v;
311     sv = null;
312 }
313         : #(VAR_LIST     { v = new Vector(); }
314               (s:ID      { v.add( s.getText() ); }
315               )*
316           )              { sv = ipt.convertVarList( v ); }
317         ;
```

```java
/*
    This is the skyEM compier
    Author : Yonghan Kim
*/

import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.debug.misc.ASTFrame;

public class skyEM  {
  public static void main(String args[]) {
    try {

      FileInputStream s = new FileInputStream(new File(args[0]));
      skyEMLexer lexer = new skyEMLexer(s);
      skyEMParser parser = new skyEMParser(lexer);
      parser.program();

      // Get the AST from the parser
      CommonAST parseTree = (CommonAST)parser.getAST();

      // Print the AST in a human-readable format
      //System.out.println(parseTree.toStringList());

      // Open a window in which the AST is displayed graphically
      ASTFrame frame = new ASTFrame("AST from the Simp parser", parseTree);
      frame.setVisible(true);

    skyEMWalker walker = new skyEMWalker();

    walker.program(parseTree);

    } catch(Exception e) { System.err.println("Exception: "+e); }
  }
}

```

```java
/*
    Boolean data type
    Author : Yonghan Kim
*/

import java.io.PrintWriter;

class skyEMBool extends skyEMType
{
    boolean var;
    skyEMBool( boolean var )
    {
        this.var = var;
    }
    public String typename()
    {
        return "bool";
    }
    public skyEMType copy()
    {
        return new skyEMBool( var );
    }
    public void print()
    {
        System.out.print( var ? "true" : "false" );
    }

    public void println()
    {
        System.out.println( var ? "true" : "false" );
    }

    public skyEMType and( skyEMType b )
    {
        if ( b instanceof skyEMBool )
            return new skyEMBool( var && ((skyEMBool)b).var );
        return error( b, "and" );
    }
    public skyEMType or( skyEMType b )
    {
        if ( b instanceof skyEMBool )
            return new skyEMBool( var || ((skyEMBool)b).var );
        return error( b, "or" );
    }
    public skyEMType not()
    {
        return new skyEMBool( !var );
    }
    public skyEMType eq( skyEMType b )
    {
        // not exclusive or
        if ( b instanceof skyEMBool )
            return new skyEMBool( ( var && ((skyEMBool)b).var ) || ( !var && !((skyEMBool)b).
        return error( b, "==" );
    }
}
```

```java
/*
    int data type
    Author : Yonghan Kim
*/
import java.io.PrintWriter;

class skyEMInt extends skyEMType
{
    int var;
    public skyEMInt( int x )
    {
        var = x;
    }
    public String typename()
    {
        return "int";
    }
    public skyEMType copy()
    {
        return new skyEMInt( var );
    }
    public static int intValue( skyEMType b )
    {
        if ( b instanceof skyEMDouble )
            return (int) ((skyEMDouble)b).var;
        if ( b instanceof skyEMInt )
            return ((skyEMInt)b).var;
        b.error( "cast to int" );
        return 0;
    }
    public void print()
    {
        System.out.print( Integer.toString( var ) );
    }

    public void println()
    {
        System.out.println( Integer.toString( var ) );
    }

    public int to_int()
    {
        return var;
    }

    public skyEMType plus( skyEMType b )
    {
        if ( b instanceof skyEMInt )
            return new skyEMInt( var + intValue(b) );
        return new skyEMDouble( var + skyEMDouble.doubleValue(b) );
    }
    public skyEMType minus( skyEMType b )
    {
        if ( b instanceof skyEMInt )
            return new skyEMInt( var - intValue(b) );
        return new skyEMDouble( var - skyEMDouble.doubleValue(b) );
    }
    public skyEMType mult( skyEMType b )
    {
        if ( b instanceof skyEMInt )
            return new skyEMInt( var * intValue(b) );
        return new skyEMDouble( var * skyEMDouble.doubleValue(b) );
    }
    public skyEMType div( skyEMType b )
    {
        if ( b instanceof skyEMInt )
            return new skyEMInt( var / intValue(b) );
        return new skyEMDouble( var / skyEMDouble.doubleValue(b) );
    }
    public skyEMType mod( skyEMType b )
    {
        if ( b instanceof skyEMInt )
            return new skyEMInt( var % intValue(b) );
        return new skyEMDouble( var % skyEMDouble.doubleValue(b) );
    }
    public skyEMType gt( skyEMType b )
```

```
 77      {
 78          if ( b instanceof skyEMInt )
 79              return new skyEMBool( var > intValue(b) );
 80          return b.lt( this );
 81      }
 82      public skyEMType ge( skyEMType b )
 83      {
 84          if ( b instanceof skyEMInt )
 85              return new skyEMBool( var >= intValue(b) );
 86          return b.le( this );
 87      }
 88      public skyEMType lt( skyEMType b )
 89      {
 90          if ( b instanceof skyEMInt )
 91              return new skyEMBool( var < intValue(b) );
 92          return b.gt( this );
 93      }
 94      public skyEMType le( skyEMType b )
 95      {
 96          if ( b instanceof skyEMInt )
 97              return new skyEMBool( var <= intValue(b) );
 98          return b.ge( this );
 99      }
100      public skyEMType eq( skyEMType b )
101      {
102          if ( b instanceof skyEMInt )
103              return new skyEMBool( var == intValue(b) );
104          return b.eq( this );
105      }
106      public skyEMType ne( skyEMType b )
107      {
108          if ( b instanceof skyEMInt )
109              return new skyEMBool( var != intValue(b) );
110          return b.ne( this );
111      }
112  }
113
```

```java
/*
    double data type
    Author : Yonghan Kim
*/

import java.io.PrintWriter;

class skyEMDouble extends skyEMType
{
    double var;
    public skyEMDouble( double x )
    {
        var = x;
    }
    public String typename()
    {
        return "double";
    }
    public skyEMType copy()
    {
        return new skyEMDouble( var );
    }
    public static double doubleValue( skyEMType b )
    {
        if ( b instanceof skyEMDouble )
            return ((skyEMDouble)b).var;
        if ( b instanceof skyEMInt )
            return (double) ((skyEMInt)b).var;
        b.error( "cast to double" );
        return 0;
    }

    public void print()
    {

        System.out.print( Double.toString( var ) );
    }

    public void println()
    {
        System.out.println( Double.toString( var ) );
    }

    public double to_double()
    {
        return var;
    }

    public skyEMType plus( skyEMType b )
    {
        return new skyEMDouble( var + doubleValue(b) );
    }

    public skyEMType minus( skyEMType b )
    {
        return new skyEMDouble( var - doubleValue(b) );
    }
    public skyEMType mult( skyEMType b )
    {
        return new skyEMDouble( var * doubleValue(b) );
    }
    public skyEMType div( skyEMType b )
    {
        return new skyEMDouble( var / doubleValue(b) );
    }
    public skyEMType mod( skyEMType b )
    {
        return new skyEMDouble( var % doubleValue(b) );
    }
    public skyEMType gt( skyEMType b )
    {
        return new skyEMBool( var > doubleValue(b) );
    }
    public skyEMType ge( skyEMType b )
    {
        return new skyEMBool( var >= doubleValue(b) );
```

```
77      }
78      public skyEMType lt( skyEMType b )
79      {
80          return new skyEMBool( var < doubleValue(b) );
81      }
82      public skyEMType le( skyEMType b )
83      {
84          return new skyEMBool( var <= doubleValue(b) );
85      }
86      public skyEMType eq( skyEMType b )
87      {
88          return new skyEMBool( var == doubleValue(b) );
89      }
90      public skyEMType ne( skyEMType b )
91      {
92          return new skyEMBool( var != doubleValue(b) );
93      }
94  }
95
```

```java
/*
    string data type
    Author : Yonghan Kim
*/
import java.io.PrintWriter;

class skyEMString extends skyEMType
{
    String var;
    public skyEMString( String str )
    {
        this.var = str;
    }
    public String typename()
    {
        return "string";
    }
    public skyEMType copy()
    {
        return new skyEMString( var );
    }

    public void print()
    {
        System.out.print( var );
    }

    public void println()
    {
        System.out.println( var );
    }

    public skyEMType plus( skyEMType b )
    {
        if ( b instanceof skyEMString )
            return new skyEMString( var + ((skyEMString)b).var );
        return error( b, "+" );
    }

    public String toString()
    {
        return var;
    }
}
```

```java
import java.io.PrintWriter;

/*
    wrapper class for undefined variables
    Author : Yonghan Kim
*/

class skyEMVariable extends skyEMType {
    public skyEMVariable( String name ) {
        super( name );
    }

    public String typename() {
        return "undefined-variable";
    }

    public skyEMType copy() {
        throw new skyEMException( "Variable " + name + " has not been defined" );
    }

    public void print() {
        System.out.println( name + " = <undefined>" );
    }
}
```

```java
/*
    Exception class
    Author : Yonghan Kim
*/

class skyEMException extends RuntimeException
{
    skyEMException( String msg )
    {
        System.err.println( "Error : " + msg );
    }
}
```

```java
import java.io.PrintWriter;
import antlr.collections.AST;

/*
    Defines the function datatype
    Author : Yonghan Kim
*/

class skyEMFunction extends skyEMType {
    // we need a reference to the AST for the function entry
    String[] args;
    AST body;
    skyEMSymbolTable pst;   // the symbol table


    public skyEMFunction( String name, String[] args,
                        AST body, skyEMSymbolTable pst) {
        super( name );
        this.args = args;
        this.body = body;
        this.pst = pst;
    }

    public String typename() {
        return "function";
    }

    public skyEMType copy() {
        return new skyEMFunction( name, args, body, pst );
    }

    public String[] getArgs() {
        return args;
    }

    public skyEMSymbolTable getParentSymbolTable() {
        return pst;
    }

    public AST getBody() {
        return body;
    }
}

```

```java
import java.util.*;
import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.RecognitionException;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;

/*
    This interpreter rountines are called from the tree walker
    Author : Yonghan Kim
*/

class skyEMInterpreter {
    skyEMSymbolTable symt;

    final static int fc_none = 0;
    final static int fc_break = 1;
    final static int fc_continue = 2;
    final static int fc_return = 3;

    private int control = fc_none;
    private String label;

    private Random random = new Random();

    public skyEMInterpreter() {
        symt = new skyEMSymbolTable( null, null );
    }

    public skyEMType[] convertExprList( Vector v ) {
        skyEMType[] x = new skyEMType[v.size()];
        for ( int i=0; i<x.length; i++ )
            x[i] = (skyEMType) v.elementAt( i );
        return x;
    }

    public static String[] convertVarList( Vector v ) {
        String[] sv = new String[ v.size() ];
        for ( int i=0; i<sv.length; i++ )
            sv[i] = (String) v.elementAt( i );
        return sv;
    }

    public static skyEMType getNumber( String s ) {
        if ( s.indexOf( '.' ) >= 0
             || s.indexOf( 'e' ) >= 0 || s.indexOf( 'E' ) >= 0 )
            return new skyEMDouble( Double.parseDouble( s ) );
        return new skyEMInt( Integer.parseInt( s ) );
    }

    public skyEMType getVariable( String s ) {

        skyEMType x = symt.getValue( s, true, 0 );
        if ( null == x )
            return new skyEMVariable( s );
        return x;
    }

    public skyEMType rvalue( skyEMType a ) {
        if ( null == a.name )
            return a;
        return a.copy();
    }

    public skyEMType deepRvalue( skyEMType a ) {
        if ( null == a.name )
            return a;
        return a.copy();
    }

    public skyEMType assign( skyEMType a, skyEMType b ) {
        if ( null != a.name )
        {

            skyEMType x = deepRvalue( b );
```

```
77              x.setName( a.name );
78              symt.setValue( x.name, x, true, 0 );
79              return x;
80          }
81          else
82          {
83
84              return a.error( b, "=" );
85          }
86      }
87
88      public skyEMType funcInvoke(
89          skyEMWalker walker,  skyEMType func,
90          skyEMType[] params ) throws antlr.RecognitionException {
91
92          // func must exist
93          if ( !( func instanceof skyEMFunction ) )
94              return func.error( "Not a function" );
95
96          // check parameters
97          String[] args = ((skyEMFunction)func).getArgs();
98          if ( args.length != params.length )
99              return func.error( "unmatched length of parameters" );
100
101         // create a new symbol table
102         symt = new skyEMSymbolTable( ((skyEMFunction)func).getParentSymbolTable(),
103                                      symt );
104
105         // assign actual parameters to formal arguments
106         for ( int i=0; i<args.length; i++ )
107         {
108             skyEMType d = rvalue( params[i] );
109             d.setName( args[i] );
110             symt.setValue( args[i], d, false, 0 );
111         }
112
113         // call the function body
114         skyEMType r = walker.expr( ((skyEMFunction)func).getBody() );
115
116         // no break or continue can go through the function
117         if ( control == fc_break || control == fc_continue )
118             throw new skyEMException( "nowhere to break or continue" );
119
120         // if a return was called
121         if ( control == fc_return )
122             tryResetFlowControl( ((skyEMFunction)func).name );
123
124         // remove this symbol table and return
125         symt = symt.dynamicParent();
126
127         return r;
128     }
129
130
131     public void funcRegister( String name, String[] args, AST body ) {
132         symt.put( name, new skyEMFunction( name, args, body, symt ) );
133     }
134
135     public void setBreak( String label ) {
136         this.label = label;
137         control = fc_break;
138     }
139
140     public void setContinue( String label ) {
141         this.label = label;
142         control = fc_continue;
143     }
144
145     public void setReturn( String label ) {
146         this.label = label;
147         control = fc_return;
148     }
149
150     public void tryResetFlowControl( String loop_label ) {
151         if ( null == label || label.equals( loop_label ) )
152             control = fc_none;
```

```
153     }
154
155     public void loopNext( String loop_label ) {
156         if ( control == fc_continue )
157             tryResetFlowControl( loop_label );
158     }
159
160     public void loopEnd( String loop_label ) {
161         if ( control == fc_break )
162             tryResetFlowControl( loop_label );
163     }
164
165     public boolean canProceed() {
166         return control == fc_none;
167     }
168 }
169
```

```java
import java.util.*;
import java.io.PrintWriter;

/*
    This is the symbol table
    Author : Yonghan Kim
*/

class skyEMSymbolTable extends HashMap {
    skyEMSymbolTable static_parent, dynamic_parent;
    boolean read_only;

    public skyEMSymbolTable( skyEMSymbolTable sparent, skyEMSymbolTable dparent ) {
        static_parent = sparent;
        dynamic_parent = dparent;
        read_only = false;
    }

    public void setReadOnly() {
        read_only = true;
    }

    public final skyEMSymbolTable staticParent() {
        return static_parent;
    }

    public final skyEMSymbolTable dynamicParent() {
        return dynamic_parent;
    }

    public final skyEMSymbolTable parent( boolean is_static ) {
        return is_static ? static_parent : dynamic_parent;
    }

    public final boolean containsVar( String name ) {
        return containsKey( name );
    }

    private final skyEMSymbolTable gotoLevel( int level, boolean is_static ) {
        skyEMSymbolTable st = this;

        if ( level < 0 )
        {
            // global variable
            while ( null != st.static_parent )
                st = st.parent( is_static );
        }
        else
        {
            // local variable
            for ( int i=level; i>0; i-- )
            {
                while ( st.read_only )
                {
                    st = st.parent( is_static );
                    assert st != null;
                }

                if ( null != st.parent( is_static ) )
                    st = st.parent( is_static );
                else
                    break;
            }
        }

        return st;
    }

    public final skyEMType getValue( String name, boolean is_static,
                                     int level ) {
        skyEMSymbolTable st = gotoLevel( level, is_static );
        Object x = st.get( name );

        while ( null == x && null != st.parent( is_static ) )
        {
            st = st.parent( is_static );
```

```
77              x = st.get( name );
78          }
79
80          return (skyEMType) x;
81      }
82
83      public final void setValue( String name, skyEMType data,
84                                  boolean is_static, int level ) {
85
86          skyEMSymbolTable st = gotoLevel( level, is_static );
87          while ( st.read_only )
88          {
89              st = st.parent( is_static );
90              assert st != null;
91          }
92
93          st.put( name, data );
94      }
95
96  }
97
```

```
1  import java.util.*;
2  import java.io.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.net.*;
6  import javax.swing.*;
7  import javax.swing.event.*;
8  import java.lang.Thread;
9
10 /*
11 ** This is Subjects class which is the main class for our program.
12 ** it constructs subjects object, which contains all the forms of
13 ** of Questions.
14 ** Author: SeungJin Nam - sn2119@columbia.edu
15 */
16
17
18 public class Subjects
19 {
20       int hour;
21       int min;
22       int sec;
23           int hour2, min2, sec2;
24       static JFrame frame;
25       javax.swing.Timer timer;
26           TimerTest timerTest;
27
28       JLabel lbPresent;
29
30         setMultipleChoice [] m = new setMultipleChoice[50];
31         setShortAnswers [] sa = new setShortAnswers[50];
32         setMatching [] ma = new setMatching[50];
33         setTrueFalse [] TF = new setTrueFalse[50];
34         String[] lists = new String[50];
35         String[] a_list;
36         int time;
37         int stopped;
38         int c_m, c_sa, c_ma, c_TF,c_a;
39         double point, total;
40         String name, id;
41         static Thread t, mc, mt, s_run, copy;
42         TimerTest w;
43
44
45         //constructor
46         public Subjects(String s1, String s2, int t)
47         {
48                 time = t;
49                 id = s1;
50                 name = s2;
51                 c_m = 0;
52                 c_sa = 0;
53                 c_ma = 0;
54                 c_TF = 0;
55                 c_a = 0;
56                 total = 0;
57                 point = 0.0;
58         }
59
60         public void setTime(int t)
61         {
62                 time = t;
63         }
64
65         public String getID()
66         {
67                 return id;
68         }
69
70         // for various types of questions
71         public void MultipleChoice()
72         {
73                 if(c_m >= m.length)
74                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
75                 if(c_a >= lists.length)
76                         lists = (String [])resizeArray(lists, 2*c_a);
```

```
77                  m[c_m] = new setMultipleChoice();
78                  lists[c_a] = "M";
79                  c_m++;
80                  c_a++;
81          }
82          public void MultipleChoice(double p)
83          {
84                  if(c_m >= m.length)
85                          m = (setMultipleChoice [])resizeArray(m, 2*c_m);
86                  if(c_a >= lists.length)
87                          lists = (String [])resizeArray(lists, 2*c_a);
88                  m[c_m] = new setMultipleChoice(p);
89                  lists[c_a] = "M";
90                  total = total + p;
91                  c_m++;
92                  c_a++;
93          }
94          public void MultipleChoice(String s, double p)
95          {
96                  if(c_m >= m.length)
97                          m = (setMultipleChoice [])resizeArray(m, 2*c_m);
98                  if(c_a >= lists.length)
99                          lists = (String [])resizeArray(lists, 2*c_a);
100                 m[c_m] = new setMultipleChoice(s,p);
101                 total = total +p;
102                 lists[c_a] = "M";
103                 c_m++;
104                 c_a++;
105         }
106         public void MultipleChoice(double p, String q, int a)
107         {
108                 if(c_m >= m.length)
109                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
110                 if(c_a >= lists.length)
111                         lists = (String [])resizeArray(lists, 2*c_a);
112                 m[c_m] = new setMultipleChoice(p, q, a);
113                 total = total + p;
114                 lists[c_a] = "M";
115                 c_m++;
116                 c_a++;
117         }
118         public void MultipleChoice(String s,double p, String q, int a)
119         {
120                 if(c_m >= m.length)
121                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
122                 if(c_a >= lists.length)
123                         lists = (String [])resizeArray(lists, 2*c_a);
124                 m[c_m] = new setMultipleChoice(s, p, q, a);
125                 lists[c_a] = "M";
126                 total = total + p;
127                 c_m++;
128                 c_a++;
129         }
130         public void MultipleChoice(double p, String q, int a, String c1, String c2, String
131         {
132                 if(c_m >= m.length)
133                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
134                 if(c_a >= lists.length)
135                         lists = (String [])resizeArray(lists, 2*c_a);
136                 m[c_m] = new setMultipleChoice(p, q, a, c1, c2, c3);
137                 lists[c_a] = "M";
138                 total = total+p;
139                 c_m++;
140                 c_a++;
141         }
142         public void MultipleChoice(String s,double p, String q, int a, String c1, String c2
143         {
144                 if(c_m >= m.length)
145                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
146                 if(c_a >= lists.length)
147                         lists = (String [])resizeArray(lists, 2*c_a);
148                 m[c_m] = new setMultipleChoice(s, p, q, a, c1, c2, c3);
149                 total = total+p;
150                 lists[c_a] = "M";
151                 c_m++;
152                 c_a++;
```

```
153         }
154         public void MultipleChoice(double p, String q, int a, String c1, String c2, String
155         {
156                 if(c_m >= m.length)
157                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
158                 if(c_a >= lists.length)
159                         lists = (String [])resizeArray(lists, 2*c_a);
160                 m[c_m] = new setMultipleChoice(p, q, a, c1, c2, c3, c4);
161                 total = total+p;
162                 lists[c_a] = "M";
163                 c_m++;
164                 c_a++;
165         }
166         public void MultipleChoice(String s, double p, String q, int a, String c1, String c
167         {
168                 if(c_m >= m.length)
169                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
170                 if(c_a >= lists.length)
171                         lists = (String [])resizeArray(lists, 2*c_a);
172                 m[c_m] = new setMultipleChoice(s, p, q, a, c1, c2, c3, c4);
173                 total = total+p;
174                 lists[c_a] = "M";
175                 c_m++;
176                 c_a++;
177         }
178         public void MultipleChoice(double p, String q, int a, String c1, String c2, String
179         {
180                 if(c_m >= m.length)
181                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
182                 if(c_a >= lists.length)
183                         lists = (String [])resizeArray(lists, 2*c_a);
184                 m[c_m] = new setMultipleChoice(p, q, a, c1, c2, c3, c4, c5);
185                 total = total+p;
186                 lists[c_a] = "M";
187                 c_m++;
188                 c_a++;
189         }
190         public void MultipleChoice(String s, double p, String q, int a, String c1, String c
191         {
192                 if(c_m >= m.length)
193                         m = (setMultipleChoice [])resizeArray(m, 2*c_m);
194                 if(c_a >= lists.length)
195                         lists = (String [])resizeArray(lists, 2*c_a);
196                 m[c_m] = new setMultipleChoice(s, p, q, a, c1, c2, c3, c4, c5);
197                 total = total+p;
198                 lists[c_a] = "M";
199                 c_m++;
200                 c_a++;
201         }
202         public void addPointMC(double p)
203         {
204                 m[c_m-1].setPoint(p);
205         }
206         public void askMC(String s)
207         {
208                 m[c_m-1].addQ(s);
209         }
210         public void answerMC(int a)
211         {
212                 m[c_m-1].addA(a);
213         }
214         public void addChoiceMC(String c1, String c2, String c3)
215         {
216                 m[c_m-1].choice(c1,c2,c3);
217         }
218         public void addChoiceMC(String c1, String c2, String c3, String c4)
219         {
220                 m[c_m-1].choice(c1,c2,c3,c4);
221         }
222         public void addChoiceMC(String c1, String c2, String c3, String c4, String c5)
223         {
224                 m[c_m-1].choice(c1,c2,c3,c4,c5);
225         }
226
227         // Now these are for the Mathing problems
228         public void Matching()
```

```
229         {
230                 if(c_ma >= ma.length)
231                         ma = (setMatching [])resizeArray(ma, 2*c_ma);
232                 if(c_a >= lists.length)
233                         lists = (String [])resizeArray(lists, 2*c_a);
234                 ma[c_ma] = new setMatching();
235                 lists[c_a] = "A";
236                 c_a++;
237                 c_ma++;
238         }
239         public void Matching(String s, double p)
240         {
241                 if(c_ma >= ma.length)
242                         ma = (setMatching [])resizeArray(ma, 2*c_ma);
243                 if(c_a >= lists.length)
244                         lists = (String [])resizeArray(lists, 2*c_a);
245                 ma[c_ma] = new setMatching(s,p);
246                 total = total+p;
247                 lists[c_a] = "A";
248                 c_a++;
249                 c_ma++;
250         }
251
252         public void addLeftSideMatching(String s)
253         {
254                 ma[c_ma-1].addLeft(s);
255         }
256         public void addRightSideMatching(String s)
257         {
258                 ma[c_ma-1].addRight(s);
259         }
260         public void addPointMatching(double p)
261         {
262                 ma[c_ma-1].setPoint(p);
263         }
264         public void MatchingSet()
265         {
266                 ma[c_ma-1].finalize();
267         }
268
269         // From here are for Short Answers
270         public void ShortAnswers()
271         {
272                 if(c_sa >= sa.length)
273                         sa = (setShortAnswers [])resizeArray(sa, 2*c_sa);
274                 if(c_a >= lists.length)
275                         lists = (String [])resizeArray(lists, 2*c_a);
276                 sa[c_sa] = new setShortAnswers();
277                 lists[c_a] = "S";
278                 c_a++;
279                 c_sa++;
280         }
281         public void ShortAnswers(double p)
282         {
283                 if(c_sa >= sa.length)
284                         sa = (setShortAnswers [])resizeArray(sa, 2*c_sa);
285                 if(c_a >= lists.length)
286                         lists = (String [])resizeArray(lists, 2*c_a);
287                 sa[c_sa] = new setShortAnswers(p);
288                 total = total+p;
289                 lists[c_a] = "S";
290                 c_a++;
291                 c_sa++;
292         }
293         public void ShortAnswers(String s, double p)
294         {
295                 if(c_sa >= sa.length)
296                         sa = (setShortAnswers [])resizeArray(sa, 2*c_sa);
297                 if(c_a >= lists.length)
298                         lists = (String [])resizeArray(lists, 2*c_a);
299                 sa[c_sa] = new setShortAnswers(s, p);
300                 total = total+p;
301                 lists[c_a] = "S";
302                 c_a++;
303                 c_sa++;
304         }
```

```
305         public void ShortAnswers(double p, String q, String a)
306         {
307                 if(c_sa >= sa.length)
308                                 sa = (setShortAnswers [])resizeArray(sa, 2*c_sa);
309                 if(c_a >= lists.length)
310                         lists = (String [])resizeArray(lists, 2*c_a);
311                 sa[c_sa] = new setShortAnswers(p, q, a);
312                 total = total+p;
313                 lists[c_a] = "S";
314                 c_a++;
315                 c_sa++;
316         }
317         public void ShortAnswers(String s, double p, String q, String a)
318         {
319                 if(c_sa >= sa.length)
320                                 sa = (setShortAnswers [])resizeArray(sa, 2*c_sa);
321                 if(c_a >= lists.length)
322                         lists = (String [])resizeArray(lists, 2*c_a);
323                 sa[c_sa] = new setShortAnswers(s, p, q, a);
324                 lists[c_a] = "S";
325                 total = total+p;
326                 c_a++;
327                 c_sa++;
328         }
329         public void addPointSA(double p)
330         {
331                 sa[c_sa-1].setPoint(p);
332         }
333         public void askSA(String q)
334         {
335                 sa[c_sa-1].addQ(q);
336         }
337         public void answerSA(String a)
338         {
339                 sa[c_sa-1].addA(a);
340         }
341
342         // From here True False
343         public void TrueFalse()
344         {
345                 if(c_TF >= TF.length)
346                         TF = (setTrueFalse [])resizeArray(TF, 2*c_TF);
347                 if(c_a >= lists.length)
348                         lists = (String [])resizeArray(lists, 2*c_a);
349                 TF[c_TF] = new setTrueFalse();
350                 lists[c_a] = "TF";
351                 c_a++;
352                 c_TF++;
353         }
354         public void TrueFalse(double p)
355         {
356                 if(c_TF >= TF.length)
357                         TF = (setTrueFalse [])resizeArray(TF, 2*c_TF);
358                 if(c_a >= lists.length)
359                         lists = (String [])resizeArray(lists, 2*c_a);
360                 TF[c_TF] = new setTrueFalse(p);
361                 total = total+p;
362                 lists[c_a] = "TF";
363                 c_a++;
364                 c_TF++;
365         }
366         public void TrueFalse(String s, double p)
367         {
368                 if(c_TF >= TF.length)
369                         TF = (setTrueFalse [])resizeArray(TF, 2*c_TF);
370                 if(c_a >= lists.length)
371                         lists = (String [])resizeArray(lists, 2*c_a);
372                 TF[c_TF] = new setTrueFalse(s, p);
373                 total = total+p;
374                 lists[c_a] = "TF";
375                 c_a++;
376                 c_TF++;
377         }
378         public void TrueFalse(double p, String a, String b)
379         {
380                 if(c_TF >= TF.length)
```

```
381                         TF = (setTrueFalse [])resizeArray(TF, 2*c_TF);
382                 if(c_a >= lists.length)
383                         lists = (String [])resizeArray(lists, 2*c_a);
384                 TF[c_TF] = new setTrueFalse(p, a, b);
385                 total = total+p;
386                 lists[c_a] = "TF";
387                 c_a++;
388                 c_TF++;
389         }
390         public void TrueFalse(String s, double p, String a, String b)
391         {
392                 if(c_TF >= TF.length)
393                         TF = (setTrueFalse [])resizeArray(TF, 2*c_TF);
394                 if(c_a >= lists.length)
395                         lists = (String [])resizeArray(lists, 2*c_a);
396                 TF[c_TF] = new setTrueFalse(s,p, a, b);
397                 total = total+p;
398                 lists[c_a] = "TF";
399                 c_a++;
400                 c_TF++;
401         }
402
403         public void addPointTF(double p)
404         {
405                 TF[c_TF-1].setPoint(p);
406         }
407         public void askTF(String q)
408         {
409                 TF[c_TF-1].addQ(q);
410         }
411         public void answerTF(String a)
412         {
413                 TF[c_TF-1].addA(a);
414         }
415
416
417
418         //this is the main method which runs the whole exam
419   public void start(String s)
420         {
421          mc = Thread.currentThread();
422          Object cin;
423          InputStreamReader in = new InputStreamReader(System.in);
424                 BufferedReader cs = new BufferedReader(in);
425          System.out.println("                                  " + name + " Exam");
426          System.out.println();
427          System.out.println("*********************************Instructions*****************
428          System.out.println("*         You have Time Limit of " + time + " minutes to compl
429          System.out.println("*       You can go back to the problem by entering \"back\" to
430          System.out.println("*        You can skip the problem by entering \"skip\" to the
431          System.out.println("*                    After the exam, in order to terminate,
432          System.out.println("*                 you will need to close all the image windows.
433          System.out.println("*****************************Press Enter to start!************
434          try{
435          cin = cs.readLine();
436          mc.sleep(1000);
437          }catch(Exception e){}
438                  javax.swing.SwingUtilities.invokeLater(new Runnable(){
439                   public void run() {
440                                       copy = Thread.currentThread();
441                                       t = copy;
442                                       int min = time, hour=0, m=0;
443                                       if(time/60 > 1 )
444                                     {
445                                             hour = time/60;
446                                             min = time % 60;
447                                     }
448                                      createAndShowGUI(0, 0, 20, mc);
449                   }
450
451                                 });
452         try{
453          mc.sleep(1000);
454          }catch(Exception e){System.out.println("Exception");}
455
456
```

```
457
458                 int prob = c_m + c_sa + c_TF + c_ma;
459                 int counter = 0, counter2 =1, forback=0;
460                 a_list = new String [c_a];
461                 int [] check0 = new int[c_m];
462                 int [] check1 = new int[c_sa];
463                 int [] check2 = new int[c_TF];
464                 int [] check3 = new int[c_ma];
465                 String [] answerlist = new String [c_a+1];
466                 String lastanswer = "a";
467                 String trigger;
468                 int c0 = 0;
469                 int c1 = 0;
470                 int c2 = 0;
471                 int c3 = 0;
472                 double point1=0;
473                 for(int j =0; j<c_sa; j++)
474                         check1[j] = -1;
475                 for(int j =0; j<c_TF; j++)
476                         check2[j] = -1;
477                 for(int j=0; j<c_m; j++)
478                         check0[j] = -1;
479                 for (int j=0; j<c_ma; j++)
480                 {
481                         check3[j] = -1;
482                 }
483                 for(int j=0; j<c_a+1; j++)
484                         answerlist[j] = "N/A";
485
486                 String back = "back";
487                 InputStreamReader stdin = new InputStreamReader(System.in);
488                 BufferedReader console = new BufferedReader(stdin);
489                 try{
490                 // in the case of ordered
491                 if(s.equals("o"))
492                 {
493                         int an = -1;
494                         String input= "null";
495                 while(prob >counter && copy.isAlive())
496                 {
497
498                         breakpoint:
499                         if(lists[counter].equals("M") && copy.isAlive())
500                         {
501                                 if(lastanswer.equals("b"))
502                                         point1 = point1 - m[c0].getPoint();
503                                 if(!(m[c0].getFilename()).equals("NULL"))
504                                 {
505                                         getImage(m[c0].getFilename());
506                                 }
507                                 System.out.println("Question " + counter2 + ". " + m[c0].ge
508                                 System.out.println("1. " + m[c0].getC1());
509                                 System.out.println("2. " + m[c0].getC2());
510                                 System.out.println("3. " + m[c0].getC3());
511                                 if (m[c0].getC4() != "false")
512                                 {
513                                         System.out.println("4. " + m[c0].getC4());
514                                 }
515                                 if (m[c0].getC5() != "false")
516                                 {
517                                         System.out.println("5. " + m[c0].getC5());
518                                 }
519                                 if(forback > counter)
520                                 {
521                                         point1 = point1 - m[c0].getPoint();
522
523                                         System.out.println("Your last answer was " + answer
524                                         //forback--;
525                                 }
526                                 System.out.print("Answer: ");
527                                 input = console.readLine();
528                                 if (input.equalsIgnoreCase(back) && counter != 0 )
529                                 {
530                                         counter--;
531                                         counter2--;
532                                         lastanswer = "b";
```

```
533                                     if(lists[counter].equals("M"))
534                                             c0--;
535                                     else if(lists[counter].equals("A"))
536                                             c1--;
537                                     else if(lists[counter].equals("S"))
538                                             c2--;
539                                     else if (lists[counter].equals("TF"))
540                                     {
541                                             c3--;
542                                     }
543                                     break breakpoint;
544                             }
545                             else if(input.equalsIgnoreCase("skip") && counter != (prob-
546                             {
547                                     counter++;
548                                     counter2++;
549                                     c0++;
550                                     break breakpoint;
551                             }
552                             lastanswer = "a";
553                             trigger = "true";
554                             while (trigger == "true")
555                             {
556                                     if(input.startsWith("1") || input.startsWith("2") |
557                                             trigger = "false";
558                                     else
559                                     {
560                                             System.out.println("please only wri
561                                             System.out.print("Answer: ");
562                                             input = console.readLine();
563                                             trigger = "true";
564                                     }
565                             }
566                             if(!input.equals("back") && counter == forback)
567                             {
568                                     answerlist[counter] = input;
569                                     forback++;
570                             }
571
572                             an = Integer.parseInt(input);
573                             if( an == m[c0].getA())
574                             {
575                                     point1 += m[c0].getPoint();
576                                     a_list[counter] = "o";
577                             }
578                             else
579                                     a_list[counter] = "x";
580                             counter++;
581                             counter2++;
582                             c0++;
583                             System.out.println();
584                     }
585                     else if (lists[counter].equals("A")&& copy.isAlive())
586                     {
587                             int j=0;
588                             if(lastanswer.equals("b"))
589                                     point1 = point1 - ma[c1].getPoint();
590
591                             System.out.println("Question " + counter2 + ". Match the fo
592                             System.out.println(ma[c1].getQuestion());
593                             Vector a = ma[c1].getQ();
594                             Object [] b = ma[c1].getQ2();
595                             Vector c = ma[c1].getA();
596                             while( j < ma[c1].getNumber())
597                             {
598                                     System.out.println(j+1 + " " +a.elementAt(j) + "\t\
599                                     j++;
600                             }
601                             j = 0;
602                             int k =1;
603                             String flag2="false";
604                             if(forback > counter)
605                             {
606                                     point1 = point1 - ma[c1].getPoint();
607                             }
608
```

```
609                             while( j < ma[c1].getNumber())
610                             {
611                                     flag2 = "false";
612                                     System.out.print("Match for " + k + "is: ");
613                                     input = console.readLine();
614                                     if (input.equalsIgnoreCase(back) && counter != 0)
615                                     {
616                                             counter--;
617                                             counter2--;
618                                             lastanswer = "b";
619                                             if(lists[counter].equals("M"))
620                                                     c0--;
621                                             else if(lists[counter].equals("A"))
622                                                     c1--;
623                                             else if(lists[counter].equals("S"))
624                                                     c2--;
625                                             else if (lists[counter].equals("TF"))
626                                             {
627                                                     c3--;
628                                             }
629                                             break breakpoint;
630                                     }
631
632                             else if(input.equalsIgnoreCase("skip") && counter != (prob-
633                             {
634                                     counter++;
635                                     counter2++;
636                                     c1++;
637                                     break breakpoint;
638                             }
639
640                                     trigger = "true";
641                             while (trigger == "true")
642                             {
643                                     if(input.startsWith("1") || input.startsWith("2") |
644                                             trigger = "false";
645                                     else
646                                     {
647                                                     System.out.println("please only wri
648                                                     System.out.print("Match for " + k +
649                                                     input = console.readLine();
650                                                     trigger = "true";
651                                     }
652                             }
653
654                                     an = Integer.parseInt(input);
655                                     trigger = "true";
656                                     while(trigger == "true")
657                                     {
658                                             if(input.startsWith("1") || input.startsWitl
659                                                     an = Integer.parseInt(input);
660                                             if(an <= b.length)
661                                             {
662                                                             if(input.startsWith("1") ||
663                                                                     trigger = "false";
664                                             }
665                                             else
666                                             {
667                                                     System.out.println("Please enter th
668                                                     System.out.print("Match for " + k +
669                                                     input = console.readLine();
670                                                     trigger = "true";
671                                             }
672                                     }
673
674                             an = Integer.parseInt(input);
675                             if(!input.equals("back") && counter == forback)
676                             {
677                                     answerlist[counter] = input;
678                                     forback++;
679                             }
680
681                                     an = Integer.parseInt(input);
682                                     Object answering = b[an-1];
683                                     Object answering2 = c.elementAt(j);
684                                     if (!answering.equals(answering2))
```

```
685                                     {
686                                             flag2 = "true";
687                                     }
688                                     j++;
689                                     k++;
690                             }
691
692                             if (flag2 == "false")
693                             {
694                                     point1 += ma[c1].getPoint();
695                                     a_list[counter] = "o";
696                             }
697                             else
698                             {
699                                     a_list[counter] = "x";
700                             }
701                             lastanswer = "a";
702                             c1++;
703                             counter++;
704                             counter2++;
705                             System.out.println();
706                     }
707                     else if(lists[counter].equals("S")&& copy.isAlive())
708                     {
709                             if(lastanswer.equals("b"))
710                                     point1 = point1 - sa[c2].getPoint();
711
712                             if(!(sa[c2].getFilename()).equals("NULL"))
713                             {
714                                     getImage(sa[c2].getFilename());
715                             }
716                             System.out.println("Question " + counter2+ ". " + sa[c2].ge
717                             if(forback > counter)
718                             {
719                                     point1 = point1 - sa[c2].getPoint();
720                                     System.out.println("Your last answer was " + answer
721                                     //forback--;
722                             }
723                             System.out.print("Answer: ");
724                             input = console.readLine();
725                             if(!input.equals("back") && counter == forback)
726                             {
727                                     answerlist[counter] = input;
728                                     forback++;
729                             }
730                             if (input.equalsIgnoreCase(back) && counter!=0)
731                             {
732                                     lastanswer = "b";
733                                     counter--;
734                                     counter2--;
735                                     if(lists[counter].equals("M"))
736                                             c0--;
737                                     else if(lists[counter].equals("A"))
738                                             c1--;
739                                     else if(lists[counter].equals("S"))
740                                             c2--;
741                                     else if (lists[counter].equals("TF"))
742                                     {
743                                             c3--;
744                                     }
745                                     break breakpoint;
746                             }
747                             else if(input.equalsIgnoreCase("skip") && counter != (prob-
748                             {
749                                     counter++;
750                                     counter2++;
751                                     c2++;
752                                     break breakpoint;
753                             }
754
755                             else if( input.equalsIgnoreCase(sa[c2].getA()))
756                             {
757                                     point1 = point1 + sa[c2].getPoint();
758                                     a_list[counter] = "o";
759                             }
760                             else
```

```
761                                     a_list[counter] = "x";
762                                 lastanswer = "a";
763                                 counter++;
764                                 counter2++;
765                                 c2++;
766                                 System.out.println();
767                         }
768                         else if(lists[counter].equals("TF")&& copy.isAlive())
769                         {
770                                 if(lastanswer.equals("b"))
771                                         point1 = point1 - TF[c3].getPoint();
772
773                                 if(!(TF[c3].getFilename()).equals("NULL"))
774                                 {
775                                         getImage(TF[c3].getFilename());
776                                 }
777                                 System.out.println("Question " + counter2 + ". This is True
778                                 System.out.println(TF[c3].getQ());
779                                 if(forback > counter)
780                                 {
781                                         point1 = point1 - TF[c3].getPoint();
782                                         System.out.println("Your last answer was " + answer
783                                         //forback--;
784                                 }
785
786                                 System.out.print("Answer: ");
787                                 input = console.readLine();
788                                 trigger = "true";
789                                 while(trigger == "true")
790                                 {
791                                         if(input.equalsIgnoreCase("true") || input.equalsIg
792                                         {
793                                                 trigger = "false";
794                                         }
795                                         else
796                                         {
797                                                 System.out.println("Please answer only in t
798                                                 System.out.print("Answer: ");
799                                                 input = console.readLine();
800                                                 trigger = "true";
801                                         }
802                                 }
803                                 if(!input.equals("back") && counter == forback)
804                                 {
805                                         answerlist[counter] = input;
806                                         forback++;
807                                 }
808                                 String a = TF[c3].getA();
809                                 if (input.equalsIgnoreCase(back))
810                                 {
811                                         counter--;
812                                         counter2--;
813                                         lastanswer = "b";
814                                         if(lists[counter].equals("M"))
815                                                 c0--;
816                                         else if(lists[counter].equals("A"))
817                                                 c1--;
818                                         else if(lists[counter].equals("S"))
819                                                 c2--;
820                                         else if (lists[counter].equals("TF"))
821                                         {
822                                                 c3--;
823                                         }
824                                         break breakpoint;
825                                 }
826                                 else if(input.equalsIgnoreCase("skip") && counter != (prob-
827                                 {
828                                         counter++;
829                                         counter2++;
830                                         c3++;
831                                         break breakpoint;
832                                 }
833
834                                 else if(a.equalsIgnoreCase(input) && counter != 0)
835                                 {
836                                         point1 = point1 + TF[c3].getPoint();
```

```
                                        a_list[counter] = "o";
                                }
                                else
                                        a_list[counter] = "x";
                                counter++;
                                counter2++;
                                c3++;
                                System.out.println();
                        }
                        lastanswer = "a";

                        if(counter == prob)
                        {
                                System.out.println("Do you want to exit (Y/N)? ");
                                input = console.readLine();
                                if(input.equalsIgnoreCase("n"))
                                {
                                        counter = 0;
                                        counter2 =1;
                                        forback=0;
                                         c0 = 0;
                                         c1 = 0;
                                         c2 = 0;
                                         c3 = 0;
                                }
                                else if(input.equalsIgnoreCase("y"))
                                {
                                        if(copy.isAlive())
                                                stop();
                                        System.out.println(Thread.currentThread());
                                        System.out.println(Thread.activeCount());
                                }

                        }
                }
                }
                // random version puts the whole questions into one list first
                else if(s.equals("r")&& copy.isAlive())
                {
                        String flag = "true", input;
                        String [] r_list = new String[c_a+1];
                        int [] ro_list = new int[c_a+1];
                        while (prob > counter)
                        {
                                Random generator = new Random();
                                int random1 = generator.nextInt(4);
                                int an;
                                int r_MC;
                                flag = "true";
                        if( random1 == 0)
                        {
                                if(c_m == c0)
                                {
                                        random1 = -1;
                                        continue;
                                }
                                r_list[counter] = "M";
                                do
                                {
                                        flag = "true";
                                        r_MC = generator.nextInt(c_m);
                                        for(int i=0; i<c_m; i++)
                                        {
                                                if(check0[i] == r_MC)
                                                        flag = "false";
                                        }
                                } while(flag == "false");
                                ro_list[counter] = r_MC;
                                check0[c0] = r_MC;
                                counter++;
                                c0++;
                        }
                        else if(random1 == 1)
                        {
                                if(c_ma == c1)
                                {
```

```
913                                     random1 = -1;
914                                     continue;
915                             }
916                             r_list[counter] = "A";
917                             do
918                             {
919                                     flag = "true";
920                                     r_MC = generator.nextInt(c_ma);
921                                     for(int i=0; i<c_ma; i++)
922                                     {
923                                             if(check3[i] == r_MC)
924                                                     flag = "false";
925                                     }
926                             } while(flag == "false");
927                             ro_list[counter] = r_MC;
928                             check3[c1] = r_MC;
929                             counter++;
930                             c1++;
931
932                     }
933                     else if(random1 == 2)
934                     {
935                             if(c_sa == c2)
936                             {
937                                     random1 = -1;
938                                     continue;
939                             }
940
941                             r_list[counter] = "S";
942                             do
943                             {
944                                     flag = "true";
945                                     r_MC = generator.nextInt(c_sa);
946                                     for(int i=0; i<c_sa; i++)
947                                     {
948                                             if(check1[i] == r_MC)
949                                                     flag = "false";
950                                     }
951                             } while(flag == "false");
952                             ro_list[counter] = r_MC;
953                             check1[c2] = r_MC;
954                             counter++;
955                             c2++;
956
957                     }
958                     else if(random1 == 3)
959                     {
960                             if(c_TF == c3)
961                             {
962                                     random1 = -1;
963                                     continue;
964                             }
965                             r_list[counter] = "TF";
966                             do
967                             {
968                                     flag = "true";
969                                     r_MC = generator.nextInt(c_TF);
970                                     for(int i=0; i<c_TF; i++)
971                                     {
972                                             if(check2[i] == r_MC)
973                                                     flag = "false";
974                                     }
975                             } while(flag == "false");
976                             ro_list[counter] = r_MC;
977                             check2[c3] = r_MC;
978                             counter++;
979                             c3++;
980
981                     }
982             }
983
984             //the actual cover through problems
985                     int an, or;
986                     counter = 0;
987                     counter2 =1;
988             while(prob > counter && copy.isAlive())
```

```
989                 {
990                         breakpoint:
991                         if(r_list[counter].equals("M"))
992                         {
993                                 int i;
994                                 or = ro_list[counter];
995
996                                 if(lastanswer.equals("b"))
997                                         point1 = point1 - m[or].getPoint();
998
999                                 if(!(m[or].getFilename()).equals("NULL"))
1000                                {
1001                                        getImage(m[or].getFilename());
1002                                }
1003                                System.out.println("Question " + counter2 + ". " + m[or].ge
1004                                System.out.println("1. " + m[or].getC1());
1005                                System.out.println("2. " + m[or].getC2());
1006                                System.out.println("3. " + m[or].getC3());
1007                                if (m[or].getC4() != "false")
1008                                {
1009                                        System.out.println("4. " + m[or].getC4());
1010                                }
1011                                if (m[or].getC5() != "false")
1012                                {
1013                                        System.out.println("5. " + m[or].getC5());
1014                                }
1015                                if(forback > counter)
1016                                {
1017                                        point1 = point1 - m[or].getPoint();
1018                                        System.out.println("Your last answer was " + answer
1019                                        //forback--;
1020                                }
1021
1022                                System.out.print("Answer: ");
1023                                input = console.readLine();
1024                                if (input.equalsIgnoreCase(back) && counter != 0)
1025                                {
1026                                        counter--;
1027                                        counter2--;
1028                                        break breakpoint;
1029                                }
1030                                else if(input.equalsIgnoreCase("skip") && counter != (prob-
1031                                {
1032                                        counter++;
1033                                        counter2++;
1034                                        break breakpoint;
1035                                }
1036
1037                                lastanswer = "a";
1038                                trigger = "true";
1039                                while (trigger == "true")
1040                                {
1041                                        if(input.startsWith("1") || input.startsWith("2") |
1042                                                trigger = "false";
1043                                        else
1044                                        {
1045                                                        System.out.println("please only wri
1046                                                        System.out.print("Answer: ");
1047                                                        input = console.readLine();
1048                                                        trigger = "true";
1049                                        }
1050                                }
1051
1052                                if(!input.equals("back") && counter == forback)
1053                                {
1054                                        answerlist[counter] = input;
1055                                        forback++;
1056                                }
1057
1058                                an = Integer.parseInt(input);
1059                                if( an == m[or].getA())
1060                                {
1061                                        point1 += m[or].getPoint();
1062                                        a_list[counter] = "o";
1063                                }
1064                                else
```

```
1065                                    a_list[counter] = "x";
1066                            counter++;
1067                            counter2++;
1068                            System.out.println();
1069                    }
1070                    else if (r_list[counter].equals("A")&& copy.isAlive())
1071                    {
1072                            int r_MC, i;
1073                            or = ro_list[counter];
1074
1075                            if(lastanswer.equals("b"))
1076                                    point1 = point1 - ma[or].getPoint();
1077
1078                            int j=0;
1079                            System.out.println("Question " + counter2 + ". Match the fo
1080                            System.out.println("Question " + counter2 + ". Match the fo
1081                            System.out.println(ma[or].getQuestion());
1082
1083                            Vector a = ma[or].getQ();
1084                            Object [] b = ma[or].getQ2();
1085                            Vector c = ma[or].getA();
1086                            while( j < ma[or].getNumber())
1087                            {
1088                                    System.out.println(j+1 + " " +a.elementAt(j) + "\t\
1089                                    j++;
1090                            }
1091                            j = 0;
1092                            int k =1;
1093                            String flag2="false";
1094                            if(forback > counter)
1095                            {
1096                                    point1 = point1 - ma[or].getPoint();
1097                                    //forback--;
1098                            }
1099
1100
1101                            while( j < ma[or].getNumber())
1102                            {
1103                                    flag2 = "false";
1104                                    System.out.print("Match for " + k + "is: ");
1105                                    input = console.readLine();
1106                                    if (input.equalsIgnoreCase(back) && counter != 0)
1107                                    {
1108                                    counter--;
1109                                    counter2--;
1110                                    break breakpoint;
1111                                    }
1112                            else if(input.equalsIgnoreCase("skip") && counter != (prob-
1113                            {
1114                                    counter++;
1115                                    counter2++;
1116                                    break breakpoint;
1117                            }
1118
1119                            trigger = "true";
1120                            while (trigger == "true")
1121                            {
1122                                    if(input.startsWith("1") || input.startsWith("2") |
1123                                            trigger = "false";
1124                                    else
1125                                    {
1126                                                    System.out.println("please only wri
1127                                                    System.out.print("Match for " + k +
1128                                                    input = console.readLine();
1129                                                    trigger = "true";
1130                                    }
1131                            }
1132
1133                                    an = Integer.parseInt(input);
1134                                    trigger = "true";
1135                                    while(trigger == "true")
1136                                    {
1137                                            if(input.startsWith("1") || input.startsWit
1138                                                    an = Integer.parseInt(input);
1139                                            if(an <= b.length)
1140                                            {
```

```
                                                        if(input.startsWith("1") ||
                                                                trigger = "false";
                                        }
                                        else
                                        {
                                                System.out.println("Please enter th
                                                System.out.print("Match for " + k +
                                                input = console.readLine();
                                                trigger = "true";
                                        }
                                }
                                lastanswer = "a";
                                an = Integer.parseInt(input);
                                Object answering = b[an-1];
                                Object answering2 = c.elementAt(j);
                                if (!answering.equals(answering2))
                                {
                                        flag2 = "true";
                                }
                                j++;
                                k++;
                        }

                        if (flag2 == "false")
                        {
                                point1 += ma[or].getPoint();
                                a_list[counter] = "o";
                        }
                        else
                                a_list[counter] = "x";
                        counter++;
                        counter2++;
                        System.out.println();
                }
                else if(r_list[counter].equals("S")&& copy.isAlive())
                {
                        int r_MC, i;
                        or = ro_list[counter];

                        if(lastanswer.equals("b"))
                                point1 = point1 - sa[or].getPoint();
                        if(!(sa[or].getFilename()).equals("NULL"))
                        {
                                getImage(sa[or].getFilename());
                        }
                        System.out.println("Question " + counter2+ ". " + sa[or].ge
                        if(forback > counter)
                        {
                                point1 = point1 - sa[or].getPoint();
                                System.out.println("Your last answer was " + answer
                                //forback--;
                        }

                        System.out.print("Answer: ");
                        input = console.readLine();
                        if(!input.equals("back") && counter == forback)
                        {
                                answerlist[counter] = input;
                                forback++;
                        }
                        else if(input.equalsIgnoreCase("skip") && counter != (prob-
                        {
                                counter++;
                                counter2++;
                                break breakpoint;
                        }


                        if (input.equalsIgnoreCase(back) && counter!=0)
                        {
                                counter--;
                                counter2--;
                                break breakpoint;
                        }
                        else if( input.equalsIgnoreCase(sa[or].getA()))
                        {
```

```
1217                                    point1 = point1 + sa[or].getPoint();
1218                                    a_list[counter] = "o";
1219                            }
1220                            else
1221                                    a_list[counter] = "x";
1222                            lastanswer = "a";
1223                            counter++;
1224                            counter2++;
1225                            System.out.println();
1226                    }
1227                    else if(r_list[counter].equals("TF")&& copy.isAlive())
1228                    {
1229                            or = ro_list[counter];
1230
1231                            int r_MC, i;
1232                            if(lastanswer.equals("b"))
1233                                    point1 = point1 - TF[or].getPoint();
1234
1235                            if(!(TF[or].getFilename()).equals("NULL"))
1236                            {
1237                                    getImage(TF[or].getFilename());
1238                            }
1239                            System.out.println("Question " + counter2 + ". This is True
1240                            if(forback > counter)
1241                            {
1242                                    point1 = point1 - TF[or].getPoint();
1243                                    System.out.println("Your last answer was " + answer
1244                                    //forback--;
1245                            }
1246
1247                            System.out.println(TF[or].getQ());
1248                            System.out.print("Answer: ");
1249                            input = console.readLine();
1250                            trigger = "true";
1251                            while(trigger == "true")
1252                            {
1253                                    if(input.equalsIgnoreCase("true") || input.equalsIg
1254                                    {
1255                                            trigger = "false";
1256                                    }
1257                                    else
1258                                    {
1259                                            System.out.println("Please answer only in t
1260                                            System.out.print("Answer: ");
1261                                            input = console.readLine();
1262                                            trigger = "true";
1263                                    }
1264                            }
1265                            if(!input.equals("back") && counter == forback)
1266                            {
1267                                    answerlist[counter] = input;
1268                                    forback++;
1269                            }
1270
1271                            String a = TF[or].getA();
1272                            if (input.equalsIgnoreCase(back))
1273                            {
1274                                    counter--;
1275                                    counter2--;
1276                                    break breakpoint;
1277                            }
1278                            else if(input.equalsIgnoreCase("skip") && counter != (prob-
1279                            {
1280                                    counter++;
1281                                    counter2++;
1282                                    break breakpoint;
1283                            }
1284
1285                            else if(a.equalsIgnoreCase(input) && counter != 0)
1286                            {
1287                                    point1 = point1 + TF[or].getPoint();
1288                                    a_list[counter] = "o";
1289                            }
1290                            else
1291                                    a_list[counter] = "x";
1292                            lastanswer = "a";
```

```
1293                                    counter++;
1294                                    counter2++;
1295                                    System.out.println();
1296                            }
1297                            if(counter == prob)
1298                            {
1299                                    System.out.println("Do you want to exit (Y/N)? ");
1300                                    input = console.readLine();
1301                                    if(input.equalsIgnoreCase("n"))
1302                                    {
1303                                            counter = 0;
1304                                            counter2 =1;
1305                                            forback=0;
1306                                    }
1307                                    else if(input.equalsIgnoreCase("y"))
1308                                    {
1309                                            if(copy.isAlive())
1310                                                    stop();
1311                                            System.out.println(Thread.currentThread());
1312                                            System.out.println(Thread.activeCount());
1313                                    }
1314
1315                            }
1316
1317                    }
1318
1319                    }
1320
1321                    }catch(IOException e){System.out.println("Exception caught");}
1322                    point = point1;
1323
1324            }
1325            public void result()
1326            {
1327                    System.out.println("**********************Score Report******************
1328                    for(int i=0; i<c_a; i++)
1329                    {
1330
1331                            System.out.print("\t" + (i+1) + ". " + a_list[i] + "\t");
1332                            if((i+1)%3 == 0)
1333                                    System.out.println();
1334                    }
1335                    System.out.println();
1336                    System.out.println("************************************************************
1337                    System.out.println("Your Score is " + point + "!");
1338            }
1339            public double score()
1340            {
1341                    return point;
1342            }
1343            public double total()
1344            {
1345                    return total;
1346            }
1347
1348            // image getting function
1349            public void getImage(String s)
1350            {
1351                    Frame mf = new Frame("ExImageLoading");
1352                    mf.add(new ExImageLoading(s), BorderLayout.CENTER);
1353                    mf.addWindowListener( new WindowAdapter()
1354                    {
1355                                    public void windowClosing(WindowEvent ev)
1356                                    {
1357                                            ev.getWindow().dispose();
1358
1359                                    }
1360                    }
1361                                    );
1362
1363                    mf.setSize(650, 500);
1364                    mf.setVisible(true);
1365            }
1366
1367            // to stop the threads when the exam is done to get rid of clock
1368             public void stop ()
```

```
        {
                        mt = null;
                    Thread th1 = timerTest.stop();
                        frame.setVisible(false);
                        frame.dispose();
                        th1.stop();
                        th1 = null;

        }
                // GUI for the clock, callint timertest class
      public void createAndShowGUI(int h, int m, int s, Thread mc)
                {
                        t = mc;

            JFrame.setDefaultLookAndFeelDecorated(true);
            frame = new JFrame("TimerTest");
            frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                        timerTest = new TimerTest(h,m,s, mc, frame);

            timerTest.setOpaque(true);
            frame.setContentPane(timerTest);
            frame.pack();
            frame.setVisible(true);

      }


        /**
* Reallocates an array with a new size, and copies the contents
* of the old array to the new array.
* @param oldArray  the old array, to be reallocated.
* @param newSize   the new array size.
* @return          A new array with the same contents.
* source code from the website http://www.source-code.biz/snippets/java/3.htm
*/
        private static Object resizeArray (Object oldArray, int newSize)
        {
                int oldSize = java.lang.reflect.Array.getLength(oldArray);
                Class elementType = oldArray.getClass().getComponentType();
                Object newArray = java.lang.reflect.Array.newInstance(elementType,newSize);
                int preserveLength = Math.min(oldSize,newSize);
                if (preserveLength > 0)
                        System.arraycopy (oldArray,0,newArray,0,preserveLength);
                return newArray;
        }

}
```

```java
import java.util.*;
import java.io.*;

/* setting up matching problems. It is consists of
** question, answer, point, and both left and right
** side entries that are equal.
** Author: SeungJin Nam - sn2119@columbia.edu
*/

public class setMatching
{
    private Vector side1 = new Vector(10, 1);
    private Vector side2 = new Vector(10, 1);
    private Object [] t_side;
    private String question;
    private int counter1;
    private int counter2;
    private double point;

    public setMatching()
    {
        side1.clear();
        side2.clear();
        counter1 = 0;
        counter2 = 0;
        point = 0.0;
    }
    public setMatching(String s, double p)
    {
        question = s;
        side1.clear();
        side2.clear();
        counter1 = 0;
        counter2 = 0;
        point = p;
    }

    public void addLeft(String s)
    {
        side1.add(s);
        counter1++;
    }
    public void addRight(String s)
    {
        side2.add(s);
        counter2++;
    }
    public void setPoint(double p)
    {
        point = p;
    }

    //this method is to randomly reorganize the other side of entries
    //but at the same time keep the answers
    public void finalize()
    {
        if (counter1 != counter2)
        {
            System.out.println("Right and Left side does not have same number of entries!");
            System.exit(1);
        }
        else
        {
            t_side = new Object[side1.size()];

        }
        Random r = new Random();
        int [] temp = new int [side2.size()];
        for(int j=0; j<side2.size(); j++)
            temp[j] = -1;
        String flag = "true";
        int c = 0, rd, i;
        while(counter2 > c)
        {
            do
            {
```

```
77                  flag = "true";
78                  rd = r.nextInt(counter2);
79                  for(i=0; i<side2.size(); i++)
80                  {
81                      if(temp[i] == rd)
82                          flag = "false";
83                  }
84              }while(flag == "false");
85              t_side[c] = side2.elementAt(rd);
86              temp[c] = rd;
87              c++;
88
89          }
90      }
91      public String getQuestion()
92      {
93          return question;
94      }
95      public Vector getQ()
96      {
97          return side1;
98      }
99      public Object[] getQ2()
100     {
101         return t_side;
102     }
103     public Vector getA()
104     {
105         return side2;
106     }
107     public double getPoint()
108     {
109         return point;
110     }
111     public int getNumber()
112     {
113         return counter1;
114     }
115 }
```

```java
import java.util.*;
import java.io.*;

/*
** This class is for the Multiple Choice Questions.
** It is consisted of question, answer, points, and
** choices that ranges from 3 to 5. It could also have
** image if neede
** Author: SeungJin Nam - sn2119@columbia.edu
*/

public class setMultipleChoice
{
    private String question;
    private int answer;
    private String choice1;
    private String choice2;
    private String choice3;
    private String choice4;
    private String choice5;
    private String filename;
    private  double point;

    /* Constructors for setMultipleChoice class, No arguments to up to 5 choices argument */
    /* if you have arguments then you have to have question, point and anwer as a pair */
    public setMultipleChoice()
    {
        filename = "NULL";
        question = "NULL";
        answer = 0;
        choice1 = "NULL";
        choice2 = "NULL";
        choice3 = "NULL";
        choice4 = "NULL";
        choice5 = "NULL";
        point = 0.0;
    }
    public setMultipleChoice(double p)
    {
        filename = "NULL";
        question = "NULL";
        answer = 0;
        choice1 = "NULL";
        choice2 = "NULL";
        choice3 = "NULL";
        choice4 = "NULL";
        choice5 = "NULL";
        point = p;
    }
    public setMultipleChoice(String f, double p)
    {
        filename = f;
        question = "NULL";
        answer = 0;
        choice1 = "NULL";
        choice2 = "NULL";
        choice3 = "NULL";
        choice4 = "NULL";
        choice5 = "NULL";
        point = p;
    }
    public setMultipleChoice(String f, double p, String q, int a)
    {
        filename = f;
        point = p;
        question = q;
        answer = a;
        choice1 = "NULL";
        choice2 = "NULL";
        choice3 = "NULL";
        choice4 = "NULL";
        choice5 = "NULL";
    }
    public setMultipleChoice(double p, String q, int a)
    {
        filename = "NULL";
```

```
77          point = p;
78          question = q;
79          answer = a;
80          choice1 = "NULL";
81          choice2 = "NULL";
82          choice3 = "NULL";
83          choice4 = "NULL";
84          choice5 = "NULL";
85      }
86      public setMultipleChoice(String f, double p, String q, int a, String c1, String c2, Stri
87      {
88          filename = f;
89          point = p;
90          question = q;
91          choice1 = c1;
92          choice2 = c2;
93          choice3 = c3;
94          choice4 = "NULL";
95          choice5 = "NULL";
96          if(a <= 0 || a >4)
97          {
98              System.out.println("Answer out of bound, answer can be either 1,2 or 3");
99              System.exit(1);
100         }
101         else
102             answer = a;
103     }
104     public setMultipleChoice(double p, String q, int a, String c1, String c2, String c3)
105     {
106         filename = "NULL";
107         point = p;
108         question = q;
109         choice1 = c1;
110         choice2 = c2;
111         choice3 = c3;
112         choice4 = "NULL";
113         choice5 = "NULL";
114         if(a <= 0 || a >4)
115         {
116             System.out.println("Answer out of bound, answer can be either 1,2 or 3");
117             System.exit(1);
118         }
119         else
120             answer = a;
121     }
122     public setMultipleChoice(String f, double p, String q, int a, String c1, String c2, Stri
123     {
124         filename = f;
125         point = p;
126         question = q;
127         choice1 = c1;
128         choice2 = c2;
129         choice3 = c3;
130         choice4 = c4;
131         choice5 = "NULL";
132         if(a <= 0 || a >5)
133         {
134             System.out.println("Answer out of bound, answer can be either 1,2,3 or 4");
135             System.exit(1);
136         }
137         else
138             answer = a;
139     }
140     public setMultipleChoice(double p, String q, int a, String c1, String c2, String c3, Str
141     {
142         filename = "NULL";
143         point = p;
144         question = q;
145         choice1 = c1;
146         choice2 = c2;
147         choice3 = c3;
148         choice4 = c4;
149         choice5 = "NULL";
150         if(a <= 0 || a >5)
151         {
152             System.out.println("Answer out of bound, answer can be either 1,2,3 or 4");
```

```
153             System.exit(1);
154         }
155         else
156             answer = a;
157     }
158     public setMultipleChoice(String f, double p, String q, int a, String c1, String c2, Stri
159     {
160         filename = f;
161         point = p;
162         question = q;
163         choice1 = c1;
164         choice2 = c2;
165         choice3 = c3;
166         choice4 = c4;
167         choice5 = c5;
168         if(a <= 0 || a >6)
169         {
170             System.out.println("Answer out of bound, answer can be either 1,2,3,4 or 5");
171             System.exit(1);
172         }
173         else
174             answer = a;
175     }
176     public setMultipleChoice(double p, String q, int a, String c1, String c2, String c3, Str
177     {
178         filename = "NULL";
179         point = p;
180         question = q;
181         choice1 = c1;
182         choice2 = c2;
183         choice3 = c3;
184         choice4 = c4;
185         choice5 = c5;
186         if(a <= 0 || a >6)
187         {
188             System.out.println("Answer out of bound, answer can be either 1,2,3,4 or 5");
189             System.exit(1);
190         }
191         else
192             answer = a;
193     }
194
195     /* Built-in function for setMultipleChoice class to set variables, ask, answer, point an
196     public void addQ(String s)
197     {
198         question = s;
199     }
200     public void addA(int s)
201     {
202         if(s <= 0 || s >5)
203         {
204             System.out.println("Answer out of bound, check the answer again please");
205             System.exit(1);
206         }
207         else
208             answer = s;
209     }
210     public void choice(String c1, String c2, String c3)
211     {
212         choice1 = c1;
213         choice2 = c2;
214         choice3 = c3;
215         choice4 = "NULL";
216         choice5 = "NULL";
217     }
218     public void choice(String c1, String c2, String c3, String c4)
219     {
220         choice1 = c1;
221         choice2 = c2;
222         choice3 = c3;
223         choice4 = c4;
224         choice5 = "NULL";
225     }
226     public void choice(String c1, String c2, String c3, String c4, String c5)
227     {
228         choice1 = c1;
```

```
229         choice2 = c2;
230         choice3 = c3;
231         choice4 = c4;
232         choice5 = c5;
233     }
234     public void setPoint(double p)
235     {
236         point = p;
237     }
238
239     public String getQ()
240     {
241         return question;
242     }
243     public int getA()
244     {
245         return answer;
246     }
247     public double getPoint()
248     {
249         return point;
250     }
251     public String getC1()
252     {
253         return choice1;
254     }
255     public String getC2()
256     {
257         return choice2;
258     }
259     public String getC3()
260     {
261         return choice3;
262     }
263     public String getC4()
264     {
265         if(choice4 == "NULL")
266             return "false";
267         return choice4;
268     }
269     public String getC5()
270     {
271         if(choice5 == "NULL")
272             return "false";
273         return choice5;
274     }
275     public String getFilename()
276     {
277         return filename;
278     }
279
280 }
281
282
283
284
285
```

```java
import java.util.*;
import java.io.*;

/* This Class is for the Short answer type questions.
** it is consisted of answer, point and the answer.
** This could also have image if needed.
** Author: SeungJin Nam - sn2119@columbia.edu
*/



public class setShortAnswers
{
    private String question;
    private String answer;
    private double point;
    private String filename;

    public setShortAnswers()
    {
        filename = "NULL";
        question = "NULL";
        answer = "NULL";
        point = 0.0;
    }
    public setShortAnswers(String f, double p)
    {
        filename = f;
        question = "NULL";
        answer = "NULL";
        point = p;
    }
    public setShortAnswers(double p)
    {
        filename = "NULL";
        question = "NULL";
        answer = "NULL";
        point = p;
    }
    public setShortAnswers(String f, double p, String q, String a)
    {
        filename = f;
        question = q;
        answer = a;
        point = p;
    }
    public setShortAnswers(double p, String q, String a)
    {
        filename = "NULL";
        question = q;
        answer = a;
        point = p;
    }
    public void addQ(String s)
    {
        question = s;
    }
    public void addA(String s)
    {
        answer = s;
    }
    public void setPoint(double p)
    {
        point = p;
    }
    public String getQ()
    {
        return question;
    }
    public String getA()
    {
        return answer;
    }
    public double getPoint()
    {
        return point;
```

```
77      }
78      public String getFilename()
79      {
80          return filename;
81      }
82  }
```

```java
import java.util.*;
import java.io.*;

/* This Class is for the TrueFalse type questions.
** it is consisted of answer, point and either true
** or false answer. Ane it could have image also.
** Author: SeungJin Nam - sn2119@columbia.edu
*/

public class setTrueFalse
{
    private String question;
    private String answer;
    private double point;
    private String filename;

    public setTrueFalse()
    {
        filename = "NULL";
        question = "NULL";
        answer = "NULL";
        point = 0.0;
    }
    public setTrueFalse(String f, double p)
    {
        filename = f;
        question = "NULL";
        answer = "NULL";
        point = p;
    }
    public setTrueFalse(double p)
    {
        filename = "NULL";
        question = "NULL";
        answer = "NULL";
        point = p;
    }
    public setTrueFalse(String f, double p, String q, String a)
    {
        filename = f;
        question = q;
        if (a == "true" || a == "false")
        {
            answer = a;
        }
        else
        {
            System.out.println("TrueFalse questions answer has to be either true or false!")
            System.exit(1);
        }
        point = p;
    }
    public setTrueFalse(double p, String q, String a)
    {
        filename = "NULL";
        question = q;
        if (a == "true" || a == "false")
        {
            answer = a;
        }
        else
        {
            System.out.println("TrueFalse questions answer has to be either true or false!")
            System.exit(1);
        }
        point = p;
    }

    public void addQ(String s)
    {
        question = s;
    }
    public void addA(String s)
    {
        if (s != "true" || s != "false")
        {
```

```
77              System.out.println("TrueFalse questions answer has to be either true or false!")
78              System.exit(1);
79          }
80          else
81              answer = s;
82
83      }
84      public void setPoint(double p)
85      {
86          point = p;
87      }
88      public String getQ()
89      {
90          return question;
91      }
92      public String getA()
93      {
94          return answer;
95      }
96      public double getPoint()
97      {
98          return point;
99      }
100     public String getFilename()
101     {
102         return filename;
103     }
104 }
```

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.io.*;

/* This class is to handle Time for the exam program
** Using swing timer, it is going to count up to the
** time passed on by Subject class.
** Author: SeungJin Nam - sn2119@columbia.edu
*/
public class TimerTest extends JPanel implements ActionListener
{

        int hour;
        int min;
        int sec;
        int hour2, min2, sec2, total;
        static JFrame frame;
        Thread t, f;
        static Thread th;
        int i=0;
        javax.swing.Timer timer;
        JLabel lbPresent;

        public TimerTest()
        {
            hour =0;
            min=0;
            sec = 0;
        }

        // constructor which calls the timer
        public TimerTest(int h, int m, int s, Thread mc, JFrame fm)
        {
            hour = 0;
            min = 0;
            sec =0;
            hour2 = h;
            min2 = m;
            sec2 = s;
            frame = fm;
            f = Thread.currentThread();
            th = f;
            t = mc;
            timer = new javax.swing.Timer(1000, this);
            timer.setInitialDelay(1000);
            timer.start();
            lbPresent = new JLabel("Time   " + hour + ":" + min + ":" + sec + " passed",Labe
            add(lbPresent);
            if (i==1)
            {
                timer.stop();
                stop();
            }
        }

      public void actionPerformed(ActionEvent e) {

            boolean tf = t.isAlive();
            breakpoint:
            if(tf)
            {
                ++sec;

                if(sec == 60)
                {
                    min++;
                    sec =0;
                }
                if (min == 60)
                {
                    min=0;
                    hour++;
                }
                if(hour2 == hour && min2 == min && sec2 == sec )
```

```
77                  {
78                      i = 1;
79                      System.out.println();
80                      System.out.println("Failed to complete an exam!");
81                      stop2();
82                  }
83                  lbPresent.setText("Time   " + hour + ":" + min + ":" + sec + " passed");
84              }
85        }
86
87        // stopped used by Subject class
88        public Thread stop ()
89        {
90            if(timer.isRunning())
91            {
92              timer.stop();
93              timer = null;
94            }
95              System.out.println(timer);
96              return th;
97
98        }
99
100       // stop used by this class
101     public void stop2()
102     {
103         timer.stop();
104         timer = null;
105         frame.setVisible(false);
106         frame.dispose();
107         Thread.currentThread().stop();
108         System.out.println(Thread.currentThread());
109     }
110
111 }
```

```java
import java.awt.*;
import java.awt.event.*;
import java.net.*;

/*
** This program is just for the image loading
** Author: SeungJin Nam - sn2119@columbia.edu
*/

public class ExImageLoading extends Canvas
{
    Image imgExample = null;

    public ExImageLoading(String f)
    {
        super();
        try
        {
            imgExample = getToolkit().getImage(f);
        } catch (Exception ex) {System.err.println("Error! - No Image Available");
                                System.exit(1);}
    }

public void paint(Graphics g)
{
    if(getToolkit().prepareImage(imgExample, -1, -1, this) == false)
    {
        g.drawString("Wait for Loading", 100, 100);
    }
    else
    {
        g.drawImage(imgExample, 10, 10, this);
    }
}
}
```