Programming Language Design and Implementation Prof. Wassim Masri

EECE 434 Fall 2015

**Project Description**

In this project, you will work with ANTLR and JASMIN to create an interpreter and a compiler.

# Background Information

ANTLR is a tool for constructing recognizers, interpreters and compilers.

JASMIN is an assembler for JVM. (Assembles java byte code into machine code)

You will also need JBE to analyze your generated code. (JVM code)

This project was tested with: ANTLR 3.2, JASMIN 2.4, and JDK 1.7.

# Part 1 – Matlab Interpreter (80%)

In this part of the project, you are required to build an interpreter for a Matlab-like language. You will start by creating a grammar (T.g) to recognize the following entities.

* An expression consists of additions, multiplications, functions, parentheses etc.
* A function has the form function\_id(arg1,arg2,…arg\_n). Before calling the function, check if the function\_id is not an existing variable. In this case, if the variable is a matrix, return the corresponding element.
* A matrix has the form [a00 , a01; a10 a11]
* A sequence has the form start:end
* An assignment has the form variable = expression
* When the interpreter identifies a line that does not end with a semi-column, the value of the expression should be printed.

An example input text file with its corresponding output is shown on the next page. Assume that the index of an array/matrix starts from 0.

After constructing your grammar, you will make use of the code you created in assignment 1. Additionally, you are recommended to implement the following methods.

* static MathObject MathObject.multiply(MathObject, MathObject) – to be called when multiplying two MathObject. Internally, the method invokes the appropriate multiplication method depending on the type of the arguments (e.g. MathScalar MathMatrix).
* static MathObject MathObject.add(MathObject, MathObject) – similar to above.
* static MathObject MathObject.u\_minus(MathObject) – to be called when a unary minus precedes a MathObject.
* MathMatrix.MathMatrix(Vector<Vector<MathObject>>) – a constructor for MathMatrix that takes a Vector of Vectors of MathObjects. Dynamically, the MathObjects are MathScalars and represent the elements of the matrix. These vectors are constructed during parsing.
* static MathObject MathFunction.call(String, Vector<MathObject>) – calls the appropriate method given its name (e.g. size, disp) and a vector of arguments.
* static MathMatrix MathFunction.createSquence(MathObject start, MathObject end) – creates an array (MathMatrix) that contains all elements from start to end with increments of 1.

Additionally, you are recommended to perform the following modifications to your grammar file:

* Import HashMap and Vector in the @header section
* In the @members section:
  + Declare a HashMap to store your variable names along with their values (type MathObject)
  + Implement a method that assigns a value to a variable.
  + Implement a method that returns the value of a variable.
* Exclude new lines (\n \t \r) from the white space token so that they are not totally ignored by the lexer. Instead, declare a new token NEWLINE : (CR | LF)+

|  |  |
| --- | --- |
| A=1+1+2\*4+3\*(1+3)  Seq=1:15;  Seq  B = Seq \* A  M1=[1 2 ; 2 1]  M2=[1 0 ; 0 1];  M3= M1\*M2  M1(1,1)  M1(1,0)  size(M1,2)  size(M1)  M1  disp(M1); | A =  22.00  Ans =  1.00 2.00 3.00 4.00 5.00 6.00 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00  B =  22.00 44.00 66.00 88.00 110.00 132.00 154.00 176.00 198.00 220.00 242.00 264.00 286.00 308.00 330.00  M1 =  1.00 2.00  2.00 1.00  M3 =  1.00 2.00  2.00 1.00  Ans =  1.00  Ans =  2.00  Ans =  2.00  Ans =  2.00 2.00  Ans =  1.00 2.00  2.00 1.00  1.00 2.00  2.00 1.00 |

# Part2 – Compiler (20%)

This part of the project is **independent** from Part 1 and is not related to Matlab. In this part you will create a compiler using JASMIN and ANTLR. Use the code provided in class as a starting point.

You are required to augment the code and grammar provided to support arrays. At the end of the project, the following input should be working:

(meaning here, Jasmin grammar is supposed to give a file MyClass.j, and then we use JASMIN to work on the output JVM code to later on produce assembly)

|  |
| --- |
| a= [4 2 3 7 1];  N = 5;  min=a(0);  i=1;  while (i<N)  {  if (a(i)<min)  {  min = a(i);  }  i = i + 1;  }  print min; |

Hints:

* When assigning an array of integers to a variable, store it in a vector.
* When the notation array(index) is used, invoke the method Vector.getElement(index).(maybe instruction.add?)

# Deliverables

Your project folder containing:

* A batch file to run part1
* A batch file to run part2
* A README file describing what you implemented
* input1.txt showing the input for part1
* input2.txt showing the input for part2