CS335A Project Milestone 4

Aditya Ranjan 190068
Shorya Kumar 190818
Akansh Agrawal 180050

Tools Used: The tools utilized include Flex , Bison , Graphviz and GCC.

1) Flex (Fast Lexical Analyzer Generator) : It is a tool used for generating scanners: programs which recognize the lexical patterns in the text. In our codebase we used it to create tokens for the Java language which are utilised by the Bison. To create an AST Node for its utilisation in AST generation we made a FixedNode function with data type Node* . This basically is used in the action rule inside the pattern matching of each token to get the node generated in the required fashion.

2) Bison : It is a general purpose parser generator based on the LALR(1) grammar which forms the bottom up parser. In this we again utilised the Node* data type to create the Nodes. The action for CompilationUnit includes the building of tree using buildTree function for generating AST and the nodes included in the tree are generated along with their children in the actions of the subsequent production rules.

3) Graphviz : It is a graph visualization tool to visualize the AST. It includes two components: the language DOT (for describing the AST) and a tool called dot.

4) GCC: It refers to the GNU Compiler Collection, which we have used in the execution for codegen (as asked in the milestone 4 description)

Building
In the terminal do the following:
$ cd src
$ make

Execution
Now further follow these instructions:
$ ../bin/3ACGenerator –input ../tests/< $testfile.java$ > –output ../graph.dot
$ gcc -no-pie output.s
$ ./a.out

Note: The src folder contains the files comprising lexer, parser , 3AC, symbol_table, codegen and the Makefile. The make command above runs the Makefile and then we run the AST generator after coming out of src folder. Further while executing instructions for codegen the output.s file gets generated which is in our target language x86_64.

Our implementation supports the required command line parameters input , output, help and verbose.
−−help displays the help screen which shows the usage of the AST generator program and about the various options supported.
−−verbose provides the debugging info.
−−input is used for taking input file as already used above.
−−output is used for getting the required output as already shown above.

Note above that double dash are without any spaces. Also the tests folder contains the 10 sample JAVA programs as required.

WE ARE SUCCESSFULLY ABLE TO IMPLEMENT ALL THE BASIC FEATURES ALONG WITH MENTIONED ADVANCED/OPTIONAL FEATURES

The list for above is a s follows:

The basic features (as in the descriptions of the project given):

1) Primitive Data Types

2) Multidimensional Arrays (for basic it is : max 3D)

3) Arithmetic Operators

4) PostIncrement, PrecIncrement, PreDecrement, PostDecrement

5) Relational Operators

6) Bitwise Operators

7) Logical Operators

8) Assignment Operators

9) Ternary Operators

10) Control Flow (if-else, for loop and while)

11) Methods and method calls, including both static and non-static methods

12) Support for recursion

13) Support for println()

14) Support for classes and objects.

The Extra Functionalities Implemented atleast as follows :

1) Print Support by Strings

2) Multidimensional Arrays (which are greater than 3 dimensions as well)

3) Constructor calls

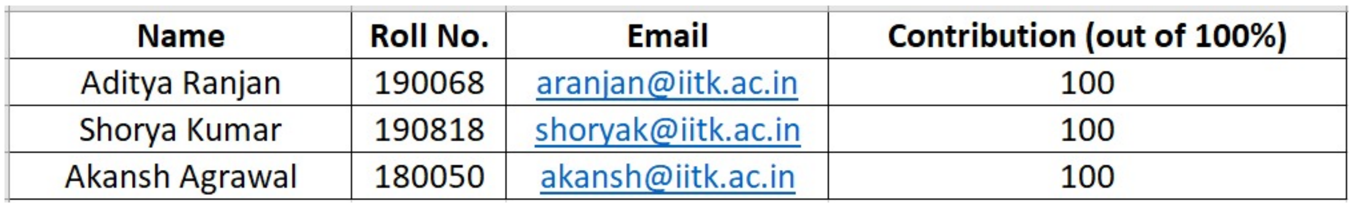
4) MultiClass Support

5) Typecasting

6) Overflow Handling

7) Object function calling

8) Break and Continue



Name	Roll No.	Email	Contribution (out of 100%)
Aditya Ranjan	190068	aranjan@iitk.ac.in	100
Shorya Kumar	190818	shoryak@iitk.ac.in	100
Akansh Agrawal	180050	akansh@iitk.ac.in	100

Figure 1: Effort Sheet