Phase3 Report

Mahmoud Abdel Latif	#64
Karim Nasr	#48
Shady Abdel Aziz	#27
Hassan Khalil	#22

Data Structures:

vector<string> codeList: to hold lines that will be printed in the output file.

typedef enum {INT_T, FLOAT_T, BOOL_T, VOID_T, ERROR_T} type_enum: to differentiate between types.

map<string, pair<int,type_enum> > symbTab: symbol table

map<string,string> inst_list: to hold instructions bytecodes

statement

```
struct {
      vector<int> *nextList;
} stmt_type;
```

expression

```
struct {
      int sType;
} expr_type;
```

boolean expression

```
struct {
      vector<int> *trueList, *falseList;
}bexpr_type;
```

Algorithms and Techniques:

-We used rules.lex to parse code to get tokens.

-Then by using bison (grammar.y) we build parse tree.

-Finally by applying semantic rules on this parse tree we generate java byte code.

Notes:

We added ‘label’ and ‘goto’ productions to some grammar rules to generate new label and adding goto statement to the code.

Main Functions:

bool checkId(string id)
to check if identifier exists in symbol table.

void var(string name, int type)
to create new variable of certain provided type, and pushing the necessary bytecode into the file.

string genNewLabel()
to generate new label.

string getLabel(int n)
to get the corresponding label of certain number

void printCode(void)
to print lines into the output file.

Assumptions:

-In arithmetic expressions, there must be a space between "-" and the next digit.

-In for loop, the second assignment must end with ";" .

How to test:

-In the project folder, open terminal and run the following:

```
$make
$./a.out <source file>
$java -jar ./jasmin-1.1/jasmin.jar bytecode.java
$java test
```