Assignment No.1

```
import java.io.*;

class SymTab {
    public static void main(String args[]) throws Exception {
        if (args.length < 1) {
            System.out.println("Please provide the input file.");
            return;
        }

        FileReader FP = new FileReader(args[0]);
        BufferedReader bufferedReader = new BufferedReader(FP);

        String line;
        int line_count = 0, LC = 0, symTabLine = 0, opTabLine = 0, litTabLine = 0,
poolTabLine = 0;

        // Data Structures
        final int MAX = 100;
        String[][] SymbolTab = new String[MAX][3];
        String[][] OpTab = new String[MAX][3];
        String[][] LitTab = new String[MAX][2];
        int[] PoolTab = new int[MAX];

        System.out.println("____________________________________________________");
        while ((line = bufferedReader.readLine()) != null) {
            line = line.trim();
            if (line.isEmpty()) {
                continue; // Skip empty lines
            }

            String[] tokens = line.split("\\s+");

            // Debugging output
            System.out.println("Line: " + line);
            System.out.println("Tokens length: " + tokens.length);
            for (int i = 0; i < tokens.length; i++) {
                System.out.println("Token[" + i + "]: " + tokens[i]);
            }

            if (line_count == 0) {
                if (tokens.length == 2 && tokens[0].equalsIgnoreCase("START")) {
                    LC = Integer.parseInt(tokens[1]); // Set LC to operand of START
                    System.out.println("LC initialized to: " + LC);
                } else {
                    System.out.println("Unexpected format for START line: " + line);
                }
            } else {
                if (tokens.length > 0 && !tokens[0].isEmpty()) {
                    // Inserting into Symbol Table
                    if (tokens.length > 1) {
                        SymbolTab[symTabLine][0] = tokens[0];
                        SymbolTab[symTabLine][1] = Integer.toString(LC);
                        SymbolTab[symTabLine][2] = Integer.toString(1);
                        symTabLine++;
                    } else {
                        System.out.println("Unexpected format for symbol line: " + line);
                    }
                } else if (tokens.length > 1 && (tokens[1].equalsIgnoreCase("DS") ||
tokens[1].equalsIgnoreCase("DC"))) {
                    // Entry into symbol table for declarative statements
                    SymbolTab[symTabLine][0] = tokens[0];
                    SymbolTab[symTabLine][1] = Integer.toString(LC);
                    SymbolTab[symTabLine][2] = tokens[2]; // Adjusted for actual length
                    symTabLine++;
                }

                if (tokens.length > 0 && tokens[0].startsWith("=")) {
                    // Entry of literals into literal table
                    LitTab[litTabLine][0] = tokens[0];
                    LitTab[litTabLine][1] = Integer.toString(LC);
                    litTabLine++;
                } else if (tokens.length > 1) {
                    // Entry of Mnemonic in opcode table
                    OpTab[opTabLine][0] = tokens[0];

                    if (tokens[0].equalsIgnoreCase("START") ||
tokens[0].equalsIgnoreCase("END") ||
                        tokens[0].equalsIgnoreCase("ORIGIN") ||
tokens[0].equalsIgnoreCase("EQU") ||
                        tokens[0].equalsIgnoreCase("LTORG")) {
                        // Assembler Directive
                        OpTab[opTabLine][1] = "AD";
                        OpTab[opTabLine][2] = "R11";
                    } else if (tokens[0].equalsIgnoreCase("DS") ||
tokens[0].equalsIgnoreCase("DC")) {
                        OpTab[opTabLine][1] = "DL";
```

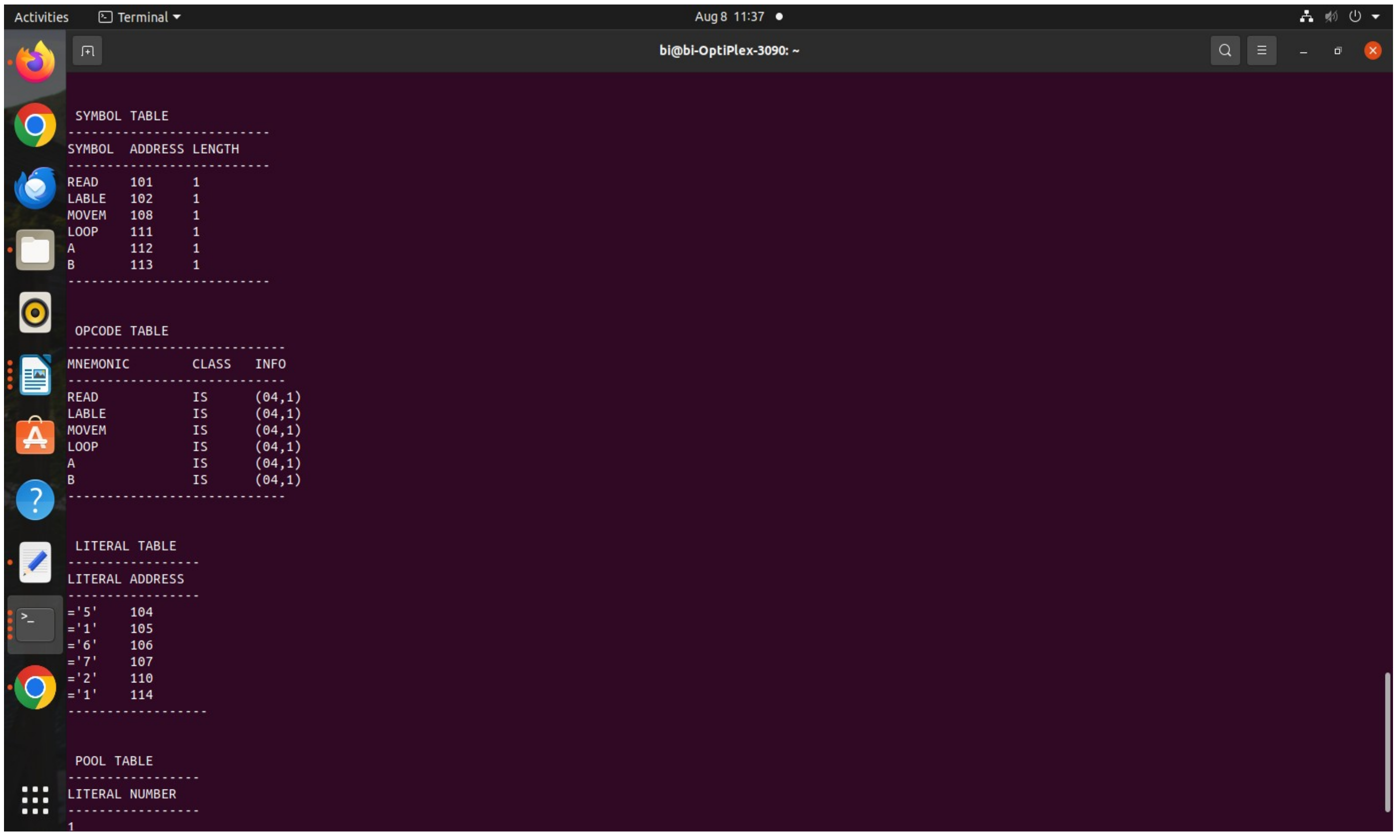
```
                        OpTab[opTabLine][2] = "R7";
                    } else {
                        OpTab[opTabLine][1] = "IS";
                        OpTab[opTabLine][2] = "(04,1)";
                    }
                    opTabLine++;
                }
            }
            line_count++;
            LC++;
        }

        System.out.println("__________________________________________________");

        // Print symbol table
        System.out.println("\n\n SYMBOL TABLE ");
        System.out.println("--------------------------");
        System.out.println("SYMBOL\tADDRESS\tLENGTH");
        System.out.println("--------------------------");
        for (int i = 0; i < symTabLine; i++)
            System.out.println(SymbolTab[i][0] + "\t" + SymbolTab[i][1] + "\t" +
    SymbolTab[i][2]);
        System.out.println("--------------------------");

        // Print opcode table
        System.out.println("\n\n OPCODE TABLE ");
        System.out.println("----------------------------");
        System.out.println("MNEMONIC\tCLASS\tINFO");
        System.out.println("----------------------------");
        for (int i = 0; i < opTabLine; i++)
            System.out.println(OpTab[i][0] + "\t\t" + OpTab[i][1] + "\t" + OpTab[i][2]);
        System.out.println("----------------------------");

        // Print literal table
        System.out.println("\n\n LITERAL TABLE ");
        System.out.println("-----------------");
        System.out.println("LITERAL\tADDRESS");
        System.out.println("-----------------");
        for (int i = 0; i < litTabLine; i++)
            System.out.println(LitTab[i][0] + "\t" + LitTab[i][1]);
        System.out.println("------------------");

        // Initialization of POOLTAB
        for (int i = 0; i < litTabLine; i++) {
            if (i + 1 < litTabLine && LitTab[i][0] != null && LitTab[i + 1][0] != null) { // If
literals are present
                if (i == 0) {
                    PoolTab[poolTabLine] = i + 1;
                    poolTabLine++;
                } else if (Integer.parseInt(LitTab[i][1]) < Integer.parseInt(LitTab[i + 1][1])
- 1) {
                    PoolTab[poolTabLine] = i + 2;
                    poolTabLine++;
                }
            }
        }

        // Print pool table
        System.out.println("\n\n POOL TABLE ");
        System.out.println("-----------------");
        System.out.println("LITERAL NUMBER");
        System.out.println("-----------------");
        for (int i = 0; i < poolTabLine; i++)
            System.out.println(PoolTab[i]);
        System.out.println("------------------");

        // Always close files.
        bufferedReader.close();
    }
}
```

input.txt:

```
START 100
READ A
LABLE MOVER A,B
LTORG
='5'
='1'
='6'
='7'
MOVEM A,B
LTORG
='2'
LOOP READ B
A DS 1
B DC '1'
='1'
```

END

Output:



```
 SYMBOL  TABLE
--------------------------
SYMBOL  ADDRESS LENGTH
--------------------------
READ    101     1
LABLE   102     1
MOVEM   108     1
LOOP    111     1
A       112     1
B       113     1
--------------------------


 OPCODE TABLE
----------------------------
MNEMONIC        CLASS   INFO
----------------------------
READ            IS      (04,1)
LABLE           IS      (04,1)
MOVEM           IS      (04,1)
LOOP            IS      (04,1)
A               IS      (04,1)
B               IS      (04,1)
----------------------------


 LITERAL  TABLE
-----------------
LITERAL ADDRESS
-----------------
='5'    104
='1'    105
='6'    106
='7'    107
='2'    110
='1'    114
------------------


 POOL  TABLE
-----------------
LITERAL NUMBER
-----------------
1
```