NAME: TEJAS NALAWADE PRACTICAL NO: 2 ROLL NO: TCOD01

Title of Assignment: Design suitable Data structures and implement Pass-II of a two-pass assembler for pseudo-machine.

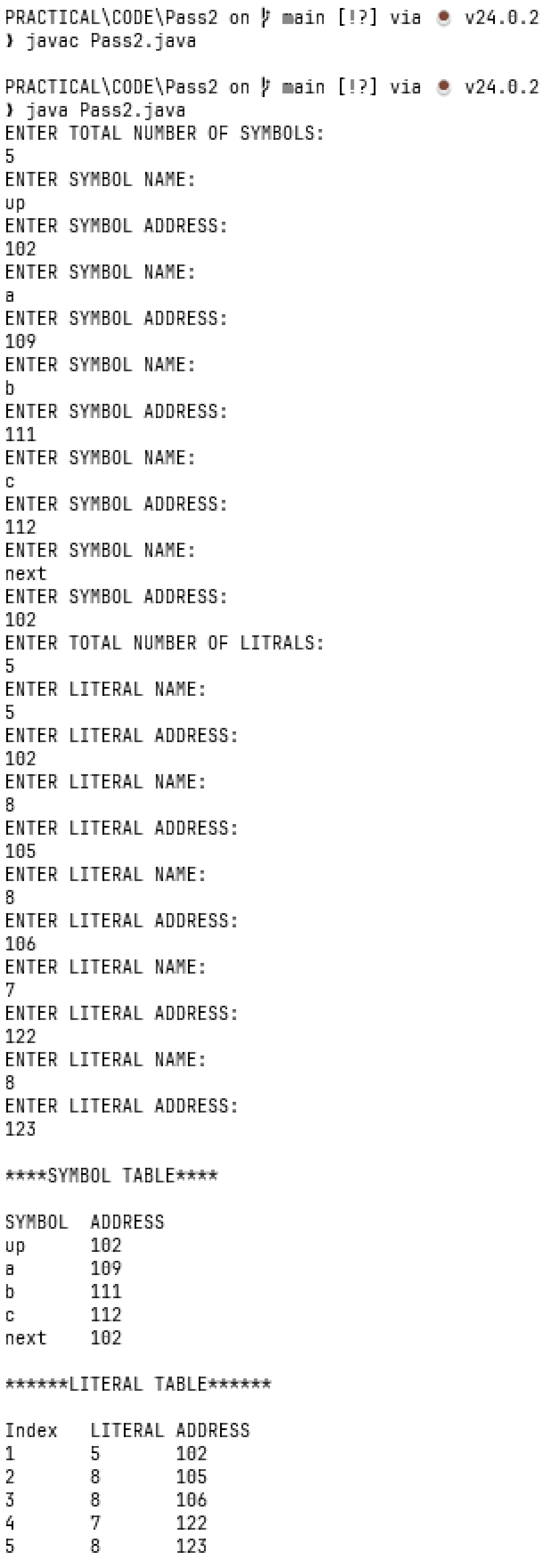
Problem Statement: Implement pass-II of TWO Pass assembler with hypothetical Instruction set using Java language. Instruction set should include all types of assembly language statements such as Imperative, Declarative and Assembler Directive. While designing stress should be given on
a) How efficiently Mnemonic opcode table could be implemented so as to enable faster retrieval on op code.
b) Implementation of symbol table, pool tables for faster retrieval.

CODE:

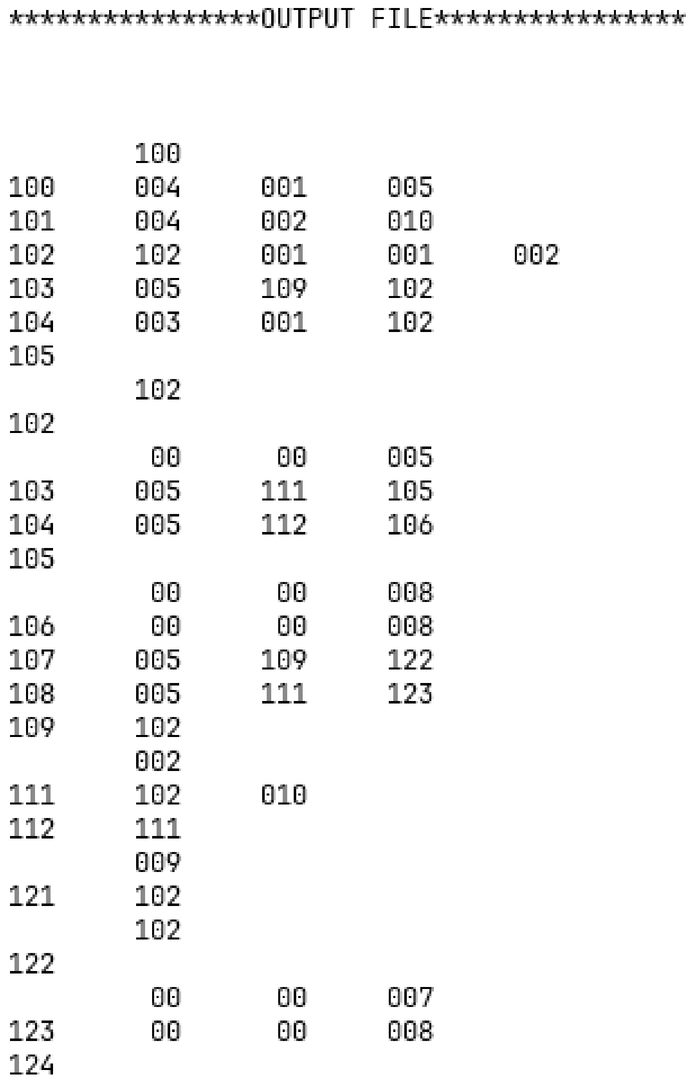
```
import java.io.*;
import java.util.Scanner;

public class Pass2 {
   static Obj[] symb_table = new Obj[10];
   static Obj[] literal_table = new Obj[10];
   static int symb_found = 0;

   public static void main(String[] args) throws IOException {
      Scanner sc = new Scanner(System.in);
      System.out.println("ENTER TOTAL NUMBER OF SYMBOLS: ");
      int total_symb = sc.nextInt();
      for (int i = 0; i < total_symb; i++) {
         symb_table[i] = new Obj("", 0);
         System.out.println("ENTER SYMBOL NAME: ");
         symb_table[i].name = sc.next();
         System.out.println("ENTER SYMBOL ADDRESS: ");
         symb_table[i].addr = sc.nextInt();
      }

      System.out.println("ENTER TOTAL NUMBER OF LITRALS: ");
      int total_ltr = sc.nextInt();
      for (int i = 0; i < total_ltr; i++) {
         literal_table[i] = new Obj("", 0);
         System.out.println("ENTER LITERAL NAME: ");
         literal_table[i].name = sc.next();
         System.out.println("ENTER LITERAL ADDRESS: ");
         literal_table[i].addr = sc.nextInt();
      }

      System.out.println("\n****SYMBOL TABLE****");
      System.out.println("\nSYMBOL\tADDRESS");
      for (int i = 0; i < total_symb; i++) {
         System.out.println(symb_table[i].name + "\t" + symb_table[i].addr);
      }

      System.out.println("\n******LITERAL TABLE******");
```

```
        System.out.println("\nIndex\tLITERAL\tADDRESS");
        for (int i = 0; i < total_ltr; i++) {
            System.out.println((i + 1) + "\t" + literal_table[i].name + "\t" + literal_table[i].addr);
        }
        BufferedReader br2 = new BufferedReader(new FileReader("Output.txt"));
        String line;
        boolean symbol_error = false, undef_mnemonic = false;
        System.out.println("\n****************OUTPUT FILE****************\n\n");

        lab:
        while ((line = br2.readLine()) != null) {
            String[] token_list = line.split("\\s+",5);
            symbol_error = false;
            undef_mnemonic = false;
            labl:
            for (String token : token_list) {
                if (token.isEmpty()) {
                    continue;
                }

                if (token.matches("[0-9]+")) {
                    System.out.print("\n" + token);
                } else if (token.startsWith("(") && token.endsWith(")")) {
                    String content = token.substring(1, token.length() - 1);

                    String[] parts = content.split(",");

                    if (parts.length == 2) {
                        String letters = parts[0].trim();
                        int num = Integer.parseInt(parts[1].trim());

    switch (letters.toUpperCase()) {
        case "S":
            if (num > 0 && num <= total_symb && symb_table[num - 1].addr != 0) {
                System.out.print("\t" + symb_table[num - 1].addr);
            } else {
                System.out.print("\t---");
                symbol_error = true;
            }
            break;
        case "L":
            if (num > 0 && num <= total_ltr) {
                System.out.print("\t" + literal_table[num - 1].addr);
            } else {
                 System.out.print("\t---");
                 symbol_error = true;
            }
            break;
        case "AD":
            System.out.print("\n");
            continue labl;
        case "DL":
            switch (num) {
                case 1:
                    System.out.print("\n");
                continue labl;
            case 2:
                System.out.print("\t 00 \t 00");
                break;
            }
            break;
        case "C":
            System.out.print(String.format("\t%03d", num));
            break;
        default:
            System.out.print(String.format("\t%03d", num));
            break;
            }

            }
            }
        }
    }
            System.out.println();

            if (symbol_error) {
            System.out.print("\n\n************SYMBOL IS NOT DEFINED*****");
            }
            if (undef_mnemonic) {
            System.out.print("\n\n****************INVALID MNEMONIC******");
            }
            int[] flag = new int[total_symb];
            for (int i = 0; i < total_symb; i++) {
            symb_found = 0;
            for (int j = 0; j < total_symb; j++) {
                if (symb_table[i].name.equalsIgnoreCase(symb_table[j].name) && flag[j] == 0) {
                    symb_found++;
                    if(symb_found > 1) flag[j] = 1;
                }
            }
            if (symb_found > 1) {
                System.out.print("\n\n*******" + symb_table[i].name + "\" IS DUPLICATE SYMBOL");
            }
            }
            br2.close();
            sc.close();
        }
    }
    class Obj {
        String name;
        int addr;

        Obj(String nm, int address) {
            this.name = nm;
            this.addr = address;
        }
    }
```

OUTPUT:



```
PRACTICAL\CODE\Pass2 on ⎇ main [!?] via ● v24.0.2
❯ javac Pass2.java

PRACTICAL\CODE\Pass2 on ⎇ main [!?] via ● v24.0.2
❯ java Pass2.java
ENTER TOTAL NUMBER OF SYMBOLS:
5
ENTER SYMBOL NAME:
up
ENTER SYMBOL ADDRESS:
102
ENTER SYMBOL NAME:
a
ENTER SYMBOL ADDRESS:
109
ENTER SYMBOL NAME:
b
ENTER SYMBOL ADDRESS:
111
ENTER SYMBOL NAME:
c
ENTER SYMBOL ADDRESS:
112
ENTER SYMBOL NAME:
next
ENTER SYMBOL ADDRESS:
102
ENTER TOTAL NUMBER OF LITRALS:
5
ENTER LITERAL NAME:
5
ENTER LITERAL ADDRESS:
102
ENTER LITERAL NAME:
8
ENTER LITERAL ADDRESS:
105
ENTER LITERAL NAME:
8
ENTER LITERAL ADDRESS:
106
ENTER LITERAL NAME:
7
ENTER LITERAL ADDRESS:
122
ENTER LITERAL NAME:
8
ENTER LITERAL ADDRESS:
123

****SYMBOL TABLE****

SYMBOL  ADDRESS
up      102
a       109
b       111
c       112
next    102

******LITERAL TABLE******

Index   LITERAL ADDRESS
1       5       102
2       8       105
3       8       106
4       7       122
5       8       123
```



```
****************OUTPUT FILE****************


        100
100     004     001     005
101     004     002     010
102     102     001     001     002
103     005     109     102
104     003     001     102
105
        102
102
         00      00     005
103     005     111     105
104     005     112     106
105
         00      00     008
106      00      00     008
107     005     109     122
108     005     111     123
109     102
        002
111     102     010
112     111
        009
121     102
        102
122
         00      00     007
123      00      00     008
124
```