NAME: TEJAS NALAWADE PRACTICAL NO: 4 ROLL NO: TCOD01

Title of Assignment: Write a Java program for pass-II of a two-pass macro-processor.

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.*;

public class Mpass2 {
   public static void main(String[] args) throws IOException {
      mdt[] MDT = new mdt[50];
      mnt[] MNT = new mnt[20];
      arglist[] formal_parameter = new arglist[20];
      arglist[] actual_parameter = new arglist[20];

      int macro_addr = -1;
      boolean macro_start = false, macro_end = false;
      int macro_call = -1;
      int mdt_cnt = 0, mnt_cnt = 0, formal_arglist_cnt = 0,
            actual_arglist_cnt = 0;

      // ---------- Read MNT ----------
      BufferedReader br1 = new BufferedReader(new FileReader("MNT.txt"));
      String line;
      while ((line = br1.readLine()) != null) {
         String[] parts = line.split("\\s+");
         if (parts.length < 3) continue;
         MNT[mnt_cnt++] = new mnt(parts[0], Integer.parseInt(parts[1]), Integer.parseInt(parts[2]));
      }
      br1.close();

      System.out.println("\n\t**********MACRO NAME TABLE**********");
      System.out.println("\n\tINDEX\tNAME\tADDRESS\tARG_CNT");
      for (int i = 0; i < mnt_cnt; i++) {
         System.out.println("\t" + i + "\t" + MNT[i].name + "\t" + MNT[i].addr + "\t" + MNT[i].arg_cnt);
      }

      // ---------- Read ARGLIST ----------
      br1 = new BufferedReader(new FileReader("ARGLIST.txt"));
      while ((line = br1.readLine()) != null) {
         String[] parameters = line.split("\\s+");
         if (parameters.length > 0) {
            formal_parameter[formal_arglist_cnt] = new arglist(parameters[0]);
            if (parameters.length > 1) {
```

```
            formal_parameter[formal_arglist_cnt].value = parameters[1];
        }
        formal_arglist_cnt++;
    }
}
br1.close();

System.out.println("\n\n\t**********FORMAL ARGUMENT LIST**********");
System.out.println("\n\tINDEX\tNAME\tVALUE");
for (int i = 0; i < formal_arglist_cnt; i++) {
    System.out.println("\t" + i + "\t" + formal_parameter[i].argname + "\t" + formal_parameter[i].value);
}

// ---------- Read MDT ----------
br1 = new BufferedReader(new FileReader("MDT.txt"));
while ((line = br1.readLine()) != null) {
    MDT[mdt_cnt] = new mdt();
    MDT[mdt_cnt++].stmnt = line;
}
br1.close();

System.out.println("\n\t**********MACRO DEFINITION TABLE**********");
System.out.println("\n\tINDEX\tSTATEMENT");
for (int i = 0; i < mdt_cnt; i++) {
    System.out.println("\t" + i + "\t" + MDT[i].stmnt);
}

// ---------- Expansion ----------
br1 = new BufferedReader(new FileReader("input.txt"));
BufferedWriter bw1 = new BufferedWriter(new FileWriter("output.txt"));

while ((line = br1.readLine()) != null) {
    line = line.replaceAll(",", " ");
    String[] tokens = line.trim().split("\\s+");

    for (String current_token : tokens) {
        if (current_token.equalsIgnoreCase("MACRO")) {
            macro_start = true;
            macro_end = false;
        } else if (current_token.equalsIgnoreCase("MEND")) {
            macro_end = true;
            macro_start = false;
        } else if (macro_end && !macro_start) {
            // check macro call
            for (int i = 0; i < mnt_cnt; i++) {
                if (current_token.equalsIgnoreCase(MNT[i].name)) {
                    macro_call = i;
                    actual_arglist_cnt = 0;
                    // collect arguments
                    for (int k = 1; k < tokens.length; k++) {
                        if (tokens[k].contains("=")) {
                            String[] pair = tokens[k].split("=");
                            actual_parameter[actual_arglist_cnt++] = new arglist(pair[1]);
                        } else {
                            actual_parameter[actual_arglist_cnt++] = new arglist(tokens[k]);
```

```
                    }
                }
                break;
            }
        }
        if (macro_call == -1) {
            bw1.write(line + "\n");
            break;
        }
    }
}

// expand macro if found
if (macro_call != -1) {
    macro_addr = MNT[macro_call].addr + 1;
    while (true) {
        if (MDT[macro_addr].stmnt.contains("MEND")) {
            macro_call = -1;
            break;
        } else {
            String[] temp_tokens = MDT[macro_addr++].stmnt.split("\\s+");
            for (String temp : temp_tokens) {
                if (temp.matches("#[0-9]+")) {
                    int num = Integer.parseInt(temp.replaceAll("[^0-9]+", ""));
                    if (actual_parameter[num - 1] != null) {
                        bw1.write(actual_parameter[num - 1].argname + "\t");
                    }
                } else {
                    bw1.write(temp + "\t");
                }
            }
            bw1.write("\n");
        }
    }
}

br1.close();
bw1.close();

System.out.println("\n\n\t*********ACTUAL ARGUMENT LIST**********");
System.out.println("\n\tINDEX\tNAME");
for (int i = 0; i < actual_arglist_cnt; i++) {
    System.out.println("\t" + i + "\t" + actual_parameter[i].argname);
}
}
}
```

OUTPUT:

```
PRACTICAL\CODE\Macro2 on  main [!?] via  v24.0.2
❯ javac Mpass2.java

PRACTICAL\CODE\Macro2 on  main [!?] via  v24.0.2
❯ java Mpass2.java

        **********MACRO NAME TABLE**********

        INDEX   NAME    ADDRESS ARG_CNT
        0       INCR    0       3
        1       DECR    5       3


        ************FORMAL ARGUMENT LIST**************

        INDEX   NAME    VALUE
        0       &x
        1       &y
        2       &REG
        3       &A
        4       &B

        ***********MACRO DEFINITION TABLE**************

        INDEX   STATEMENT
        0               INCR    &x      &y      &REG  =  AREG
        1               MOVER   #3      #1
        2               ADD         #3  #2
        3               MOVEM   #3      #1
        4               MEND
        5               DECR    &A      &B      &REG   =  BREG
        6               MOVER   #3      #4
        7               MEND


        *********ACTUAL ARGUMENT LIST**********

        INDEX   NAME
        0       N1
        1       N2
```