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:**

