**EXP 2.** Design suitable data structures and implement Pass-I and Pass-II of a two-pass macro- processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

**Main.java**

import java.util.\*;

import java.io.\*;

class Main

{

static String mnt[][]=new String[5][3]; //assuming 5 macros in 1 program

static String ala[][]=new String[10][2]; //assuming 2 arguments in each macro

static String mdt[][]=new String[20][1]; //assuming 4 LOC for each macro

static int mntc=0,mdtc=0,alac=0;

public static void main(String args[])

{

pass1();

System.out.println("\n\*\*\*\*\*\*\*\*\*PASS-1 MACROPROCESSOR\*\*\*\*\*\*\*\*\*\*\*\n");

System.out.println("MACRO NAME TABLE (MNT)\n");

System.out.println("i macro loc\n");

display(mnt,mntc,3);

System.out.println("\n");

System.out.println("ARGUMENT LIST ARRAY(ALA) for Pass1\n");

display(ala,alac,2);

System.out.println("\n");

System.out.println("MACRO DEFINITION TABLE (MDT)\n");

display(mdt,mdtc,1);

System.out.println("\n");

}

static void pass1()

{

int index=0,i;

String s,prev="",substring;

try

{

BufferedReader inp = new BufferedReader(new FileReader("input.txt"));

File op = new File("pass1\_output.txt");

if (!op.exists())

op.createNewFile();

BufferedWriter output = new BufferedWriter(new FileWriter(op.getAbsoluteFile()));

while((s=inp.readLine())!=null)

{

if(s.equalsIgnoreCase("MACRO"))

{

prev=s;

for(;!(s=inp.readLine()).equalsIgnoreCase("MEND");mdtc++,prev=s)

{

if(prev.equalsIgnoreCase("MACRO"))

{

StringTokenizer st=new StringTokenizer(s);

String str[]=new String[st.countTokens()];

for(i=0;i<str.length;i++)

str[i]=st.nextToken();

mnt[mntc][0]=(mntc+1)+""; //mnt formation

mnt[mntc][1]=str[0];

mnt[mntc++][2]=(++mdtc)+"";

st=new StringTokenizer(str[1],","); //tokenizing the arguments

String string[]=new String[st.countTokens()];

for(i=0;i<string.length;i++)

{

string[i]=st.nextToken();

ala[alac][0]=alac+""; //ala table formation

index=string[i].indexOf("=");

if(index!=-1)

ala[alac++][1]=string[i].substring(0,index);

else

ala[alac++][1]=string[i];

}

}

else //automatically eliminates tagging of arguments in definition

{ //mdt formation

index=s.indexOf("&");

substring=s.substring(index);

for(i=0;i<alac;i++)

if(ala[i][1].equals(substring))

s=s.replaceAll(substring,"#"+ala[i][0]);

}

mdt[mdtc-1][0]=s;

}

mdt[mdtc-1][0]=s;

}

else

{

output.write(s);

output.newLine();

}

}

output.close();

}

catch(FileNotFoundException ex)

{

System.out.println("UNABLE TO END FILE ");

}

catch(IOException e)

{

e.printStackTrace();

}

}

static void display(String a[][],int n,int m)

{

int i,j;

for(i=0;i<n;i++)

{

for(j=0;j<m;j++)

System.out.print(a[i][j]+" ");

System.out.println();

}

}

}

**input.txt**

MACRO

INCR1 &FIRST,&SECOND=DATA9

A 1,&FIRST

L 2,&SECOND

MEND

MACRO

INCR2 &ARG1,&ARG2=DATA5

L 3,&ARG1

ST 4,&ARG2

MEND

PRG2 START

USING \*,BASE

INCR1 DATA1

INCR2 DATA3,DATA4

FOUR DC F'4'

FIVE DC F'5'

BASE EQU 8

TEMP DS 1F

DROP 8

END

**pass1\_output.txt**

PRG2 START

USING \*,BASE

INCR1 DATA1

INCR2 DATA3,DATA4

FOUR DC F'4'

FIVE DC F'5'

BASE EQU 8

TEMP DS 1F

DROP 8

END

**Output**

