//TWO PASS MACROPROCESSOR  
import java.util.\*;  
import java.io.\*;  
class MntTuple {  
String name;  
int index;  
MntTuple(String s, int i) {  
name = s;  
index = i;  
}  
public String toString() {  
return("[" + name + ", " + index + "]");  
}  
}  
class MacroProcessor {  
static List<MntTuple> mnt;  
static List<String> mdt;  
static int mntc;  
static int mdtc;  
static int mdtp;  
static BufferedReader input;  
static List<List <String>> ala;  
static Map<String, Integer> ala\_macro\_binding;  
public static void main(String args[]) throws Exception {  
initializeTables();  
System.out.println("===== PASS 1 =====\n");  
pass1();  
System.out.println("\n===== PASS 2 =====\n");  
pass2();  
}  
static void pass1() throws Exception {  
String s = new String();  
input = new BufferedReader(new InputStreamReader(new  
FileInputStream("input.txt")));  
PrintWriter output = new PrintWriter(new  
FileOutputStream("output\_pass1.txt"), true);  
while((s = input.readLine()) != null) {  
if(s.equalsIgnoreCase("MACRO")) {  
processMacroDefinition();  
} else {  
output.println(s);  
}  
}  
System.out.println("ALA:");

showAla(1);  
System.out.println("\nMNT:");  
showMnt();  
System.out.println("\nMDT:");  
showMdt();  
}  
static void processMacroDefinition() throws Exception {  
String s = input.readLine();  
String macro\_name = s.substring(0, s.indexOf(" "));  
mnt.add(new MntTuple(macro\_name, mdtc));  
mntc++;  
pass1Ala(s);  
StringTokenizer st = new StringTokenizer(s, " ,", false);  
String x = st.nextToken();  
for(int i=x.length() ; i<12 ; i++) {  
x += " ";  
}  
String token = new String();  
int index;  
token = st.nextToken();  
x += token;  
while(st.hasMoreTokens()) {  
token = st.nextToken();  
x += "," + token;  
}  
mdt.add(x);  
mdtc++;  
addIntoMdt(ala.size()-1);  
}  
static void pass1Ala(String s) {  
StringTokenizer st = new StringTokenizer(s, " ,", false);  
String macro\_name = st.nextToken();  
List<String> l = new ArrayList<>();  
int index;  
while(st.hasMoreTokens()) {  
String x = st.nextToken();  
if((index = x.indexOf("=")) != -1) {  
x = x.substring(0, index);  
}  
l.add(x);  
}  
ala.add(l);  
ala\_macro\_binding.put(macro\_name,  
ala\_macro\_binding.size());  
}  
static void addIntoMdt(int ala\_number) throws Exception {  
String temp = new String();  
String s = new String();  
List l = ala.get(ala\_number);

boolean isFirst;  
while(!s.equalsIgnoreCase("MEND")) {  
isFirst = true;  
s = input.readLine();  
String line = new String();  
StringTokenizer st = new StringTokenizer(s, " ,",  
false);  
temp = st.nextToken();  
for(int i=temp.length() ; i<12 ; i++) {  
temp += " ";  
}  
line += temp;  
while(st.hasMoreTokens()) {  
temp = st.nextToken();  
if(temp.startsWith("&")) {  
int x = l.indexOf(temp);  
temp = ",#" + x;  
isFirst = false;  
} else if(!isFirst) {  
temp = "," + temp;  
}  
line += temp;  
}  
mdt.add(line);  
mdtc++;  
}  
}  
static void showAla(int pass) throws Exception {  
PrintWriter out = new PrintWriter(new  
FileOutputStream("out\_ala\_pass" + pass + ".txt"), true);  
for(List l : ala) {  
System.out.println(l);  
out.println(l);  
}  
}  
static void showMnt() throws Exception {  
PrintWriter out = new PrintWriter(new  
FileOutputStream("out\_mnt.txt"), true);  
for(MntTuple l : mnt) {  
System.out.println(l);  
out.println(l);  
}  
}  
static void showMdt() throws Exception {  
PrintWriter out = new PrintWriter(new  
FileOutputStream("out\_mdt.txt"), true);  
for(String l : mdt) {  
System.out.println(l);  
out.println(l);

}  
}  
static void pass2() throws Exception {  
input = new BufferedReader(new InputStreamReader(new  
FileInputStream("output\_pass1.txt")));  
PrintWriter output = new PrintWriter(new  
FileOutputStream("output\_pass2.txt"), true);  
String token = new String();  
String s;  
while((s = input.readLine()) != null) {  
StringTokenizer st = new StringTokenizer(s, " ",  
false);  
while(st.hasMoreTokens()) {  
token = st.nextToken();  
if(st.countTokens() > 2) {  
token = st.nextToken();  
}  
MntTuple x = null;  
for(MntTuple m : mnt) {  
if(m.name.equalsIgnoreCase(token)) {  
x = m;  
break;  
}  
}  
if(x != null) {  
mdtp = x.index;  
List<String> l = pass2Ala(s);  
mdtp++;  
String temp = new String();  
while(!(temp =  
mdt.get(mdtp)).trim().equalsIgnoreCase("MEND")) {  
String line = new String();  
StringTokenizer st2 = new  
StringTokenizer(temp, " ,",false);  
for(int i=0 ; i<12 ; i++) {  
line += " ";  
}  
String opcode = st2.nextToken();  
line += opcode;  
for(int i=opcode.length() ; i<24 ;  
i++) {  
line += " ";  
}  
line += st2.nextToken();  
while(st2.hasMoreTokens()) {  
String token2 = st2.nextToken();  
int index;  
if((index = token2.indexOf("#"))  
!= -1) {  
line += "," +  
l.get(Integer.parseInt(token2.substring(index+1,index+2)));

}  
}  
mdtp++;  
output.println(line);  
System.out.println(line);  
}  
break;  
} else {  
output.println(s);  
System.out.println(s);  
break;  
}  
}  
}  
System.out.println("\nALA:");  
showAla(2);  
}  
static List<String> pass2Ala(String s) {  
StringTokenizer st = new StringTokenizer(s, " ", false);  
int num\_tokens = st.countTokens();  
String macro\_name = st.nextToken();  
int ala\_no = ala\_macro\_binding.get(macro\_name);  
List<String> l = ala.get(ala\_no);  
int ctr = 0;  
StringTokenizer st2 = null;  
try {  
st2 = new StringTokenizer(st.nextToken(), ",", false);  
while(st2.hasMoreTokens()) {  
l.set(ctr, st2.nextToken());  
ctr++;  
}  
} catch(Exception e) {  
// do nothing  
}  
if(ctr < num\_tokens) {  
String s2 = mdt.get(mdtp);  
StringTokenizer st3 = new StringTokenizer(s2, " ,",  
false);  
String token = new String();  
int index = 0;  
while(st3.hasMoreTokens()) {  
token = st3.nextToken();  
if((index = token.indexOf("=")) != -1) {  
try {  
l.set(ctr++, token.substring(index+1,  
token.length()));  
} catch(Exception e) {  
// do nothing  
}  
}  
}

}  
ala.set(ala\_no, l);  
return l;  
}  
static void initializeTables() {  
mnt = new LinkedList<>();  
mdt = new ArrayList<>();  
ala = new LinkedList<>();  
mntc = 0;  
mdtc = 0;  
ala\_macro\_binding = new HashMap<>();  
}  
}  
/\*  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
INPUT  
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  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
OUTPUT  
pvgcoen-3@pvgcoen3-ThinkCentre-M700:~/PRACT4$ javac  
MacroProcessor.java  
pvgcoen-3@pvgcoen3-ThinkCentre-M700:~/PRACT4$ java MacroProcessor  
===== PASS 1 =====  
ALA:  
[&FIRST, &SECOND]  
[&ARG1, &ARG2]

MNT:  
[INCR1, 0]  
[INCR2, 4]  
MDT:  
INCR1 &FIRST,&SECOND=DATA9  
A 1,#0  
L 2,#1  
MEND  
INCR2 &ARG1,&ARG2=DATA5  
L 3,#0  
ST 4,#1  
MEND  
===== PASS 2 =====  
PRG2 START  
USING \*,BASE  
A 1,DATA1  
L 2,DATA9  
L 3,DATA3  
ST 4,DATA4  
FOUR DC F'4'  
FIVE DC F'5'  
BASE EQU 8  
TEMP DS 1F  
DROP 8  
END  
ALA:  
[DATA1, DATA9]  
[DATA3, DATA4]  
\*/

/\*

Problem Statement : Write a Java program for pass-II of a two-pass macro-processor. The output of assignment-3

(MNT, MDT and file without any macro definitions) should be input for this assignment.

\*/

import java.io.\*;

import java.util.HashMap;

import java.util.Vector;

public class macroPass2 {

public static void main(String[] Args) throws IOException{

BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));

BufferedReader b2 = new BufferedReader(new FileReader("mnt.txt"));

BufferedReader b3 = new BufferedReader(new FileReader("mdt.txt"));

BufferedReader b4 = new BufferedReader(new FileReader("kpdt.txt"));

FileWriter f1 = new FileWriter("Pass2.txt");

HashMap<Integer,String> aptab=new HashMap<Integer,String>();

HashMap<String,Integer> aptabInverse=new HashMap<String,Integer>();

HashMap<String,Integer> mdtpHash=new HashMap<String,Integer>();

HashMap<String,Integer> kpdtpHash=new HashMap<String,Integer>();

HashMap<String,Integer> kpHash=new HashMap<String,Integer>();

HashMap<String,Integer> macroNameHash=new HashMap<String,Integer>();

Vector<String>mdt=new Vector<String>();

Vector<String>kpdt=new Vector<String>();

String s,s1;

int i,pp,kp,kpdtp,mdtp,paramNo;

while((s=b3.readLine())!=null)

mdt.addElement(s);

while((s=b4.readLine())!=null)

kpdt.addElement(s);

while((s=b2.readLine())!=null){

String word[]=s.split("\t");

s1=word[0]+word[1];

macroNameHash.put(word[0],1);

kpHash.put(s1,Integer.parseInt(word[2]));

mdtpHash.put(s1,Integer.parseInt(word[3]));

kpdtpHash.put(s1,Integer.parseInt(word[4]));

}

while((s=b1.readLine())!=null){

String b1Split[]=s.split("\\s");

if(macroNameHash.containsKey(b1Split[0])){

pp= b1Split[1].split(",").length-b1Split[1].split("=").length+1;

kp=kpHash.get(b1Split[0]+Integer.toString(pp));

mdtp=mdtpHash.get(b1Split[0]+Integer.toString(pp));

kpdtp=kpdtpHash.get(b1Split[0]+Integer.toString(pp));

String actualParams[]=b1Split[1].split(",");

paramNo=1;

for(int j=0;j<pp;j++){

aptab.put(paramNo, actualParams[paramNo-1]);

aptabInverse.put(actualParams[paramNo-1],paramNo);

paramNo++;

}

i=kpdtp-1;

for(int j=0;j<kp;j++){

String temp[]=kpdt.get(i).split("\t");

aptab.put(paramNo,temp[1]);

aptabInverse.put(temp[0],paramNo);

i++;

paramNo++;

}

i=pp+1;

while(i<=actualParams.length){

String initializedParams[]=actualParams[i-1].split("=");

aptab.put(aptabInverse.get(initializedParams[0].substring(1,initializedParams[0].length())),initializedParams[1].substring(0,initializedParams[1].length()));

i++;

}

i=mdtp-1;

while(mdt.get(i).compareToIgnoreCase("MEND")!=0){

f1.write("+ ");

for(int j=0;j<mdt.get(i).length();j++){

if(mdt.get(i).charAt(j)=='#')

f1.write(aptab.get(Integer.parseInt("" + mdt.get(i).charAt(++j))));

else

f1.write(mdt.get(i).charAt(j));

}

f1.write("\n");

i++;

}

aptab.clear();

aptabInverse.clear();

}

else

f1.write("+ "+s+"\n");

}

b1.close();

b2.close();

b3.close();

b4.close();

f1.close();

}

}

/\*

OUTPUT:

neha@neha-1011PX:~/Desktop/neha\_SPOS/Turn1/A4$ javac macroPass2.java

neha@neha-1011PX:~/Desktop/neha\_SPOS/Turn1/A4$ java macroPass2

neha@neha-1011PX:~/Desktop/neha\_SPOS/Turn1/A4$ cat Pass2.txt

+ MOVE AREG,10

+ ADD AREG,='1'

+ MOVER AREG,20

+ ADD AREG,='5'

+ MOVER &AREG,100

+ MOVER &BREG,200

+ ADD &AREG,='15'

+ ADD &BREG,='10'

\*/