```
package a3;



import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.StringTokenizer;


public class MacroProcessor_PassOne {

	static List<String> MDT;

	static Map<String, String> MNT;

	static int mntPtr, mdtPtr;

	static Map<String,String> ALA;



	public static void main(String[] args) {
```

```
		try{
			pass1();
		}catch(Exception ex){
			ex.printStackTrace();
		}
	}

	static void pass1() throws Exception {

		//Initiallize data structures
		MDT = new ArrayList<String>();
		MNT = new LinkedHashMap<String, String>();
		ALA = new HashMap<String,String>();
		mntPtr =0; mdtPtr = 0;

		BufferedReader input = new BufferedReader(new InputStreamReader(new FileInputStream("C:\\Users\\Aditi\\eclipse-workspace\\atp\\src\\a3\\input.txt")));
		PrintWriter out_pass1 = new PrintWriter(new FileWriter("C:\\Users\\Aditi\\eclipse-workspace\\atp\\src\\a3\\output_pass1.txt"), true);
		PrintWriter out_mnt = new PrintWriter(new FileWriter("C:\\Users\\Aditi\\eclipse-workspace\\atp\\src\\a3\\MNT.txt"), true);
		PrintWriter out_mdt = new PrintWriter(new FileWriter("C:\\Users\\Aditi\\eclipse-workspace\\atp\\src\\a3\\MDT.txt"), true);

		String s;
		boolean processingMacroDefinition = false;
		boolean processMacroName = false;
System.out.println("NAME: Bhavika Patil");
```

```
System.out.println("ROLL NO.: TBCO22172");
System.out.println("============= Pass 1 Output ==============");
//Read from input file one line at a time
while((s = input.readLine()) != null) {
        //For each line, separate out the tokens
        String s_arr[] = tokenizeString(s," ");

        //Analyze first token to check if it is a macro definition
        String curToken = s_arr[0];
        if(curToken.equalsIgnoreCase("MACRO")){
                processingMacroDefinition = true;
                processMacroName = true;
        }
        else if(processingMacroDefinition == true){
                if(curToken.equalsIgnoreCase("MEND")){
                        MDT.add(mdtPtr++, s);
                        processingMacroDefinition = false;
                        continue;
                }
                //Insert Macro Name into MNT
                if(processMacroName == true){
                        MNT.put(curToken, mdtPtr+"");
                        mntPtr++;
                        processMacroName = false;
```

```
                        processArgumentList(s_arr[1]);
                        MDT.add(mdtPtr,s);
                        mdtPtr++;
                        continue;
                    }
                    //Convert arguments in the definition into corresponding indexed notation
                    //ADD &REG,&X == ADD #2,#1
                    String indexedArgList = processArguments(s_arr[1]);
                    MDT.add(mdtPtr++, curToken + " " + indexedArgList);
                }
                else{
                    //If line is not part of a Macro definition print the line as it is in the output file
                    System.out.println(s);
                    out_pass1.println(s);
                }
            }
            input.close();
            //Print MNT
            System.out.println("============= MNT ==============");
            Iterator<String> itMNT = MNT.keySet().iterator();
            String key, mntRow, mdtRow;
            while(itMNT.hasNext()){
                key = (String)itMNT.next();
                mntRow = key + " " + MNT.get(key);
```

```
                System.out.println(mntRow);
                out_mnt.println(mntRow);

        }
        //Print MDT
        System.out.println("============= MDT ==============");
        for(int i = 0; i < MDT.size(); i++){
                mdtRow = i + "  " + MDT.get(i);
                System.out.println(mdtRow);
                out_mdt.println(mdtRow);
        }
        out_pass1.close();
        out_mnt.close();
        out_mdt.close();
}

static void processArgumentList(String argList){
        StringTokenizer st = new StringTokenizer(argList, ",", false);
        //For each macro definition, remove contents of the HashMap
        //which are arguments from previous macro definition
        ALA.clear();
        int argCount = st.countTokens();
        //Put all arguments for current macro definition in the HashMap
        //with argument as key and argument index as value
```

```
		String curArg;
		for(int i=1 ; i <= argCount ; i++) {
			curArg = st.nextToken();
			if(curArg.contains("=")){
				curArg = curArg.substring(0,curArg.indexOf("="));
			}
			ALA.put(curArg, "#"+i);
		}
	}

	static String processArguments(String argList){
		StringTokenizer st = new StringTokenizer(argList, ",", false);
		int argCount = st.countTokens();
		String curArg, argIndexed;
		for(int i=0 ; i < argCount ; i++) {
			curArg = st.nextToken();
			argIndexed = ALA.get(curArg);
			argList = argList.replaceAll(curArg, argIndexed);
		}
		return argList;
	}

	static String[] tokenizeString(String str, String separator){
		StringTokenizer st = new StringTokenizer(str, separator, false);
```

```
                //Construct an array of the separated tokens
                String s_arr[] = new String[st.countTokens()];
                for(int i=0 ; i < s_arr.length ; i++) {
                        s_arr[i] = st.nextToken();
                }
                return s_arr;
        }


}
```