Practical No 04

```
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.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.StringTokenizer;

public class MacroProcessor_PassTwo {
    static List<String> MDT;
    static Map<String, String> MNT;
    static int mntPtr, mdtPtr;
    static List<String> formalParams, actualParams;

    public static void main(String[] args) {
        try{
            initializeTables();
            pass2();
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }

    static void pass2() throws Exception {
        BufferedReader input = new BufferedReader(new
        InputStreamReader(new FileInputStream("A4/output_pass1.txt")));
        PrintWriter out_pass2 = new PrintWriter(new
        FileWriter("A4/output_pass2.txt"), true);

        System.out.println("=========== Pass 2 Output ============");
        //Read from input file one line at a time
        String s;
        while((s = input.readLine()) != null) {
            String s_arr[] = tokenizeString(s, " ");
            //First token will either be a mnemonic or a macro call
            if(MNT.containsKey(s_arr[0])) {
                //It is a macro call
                //Create an array list of formal parameters
                String actual_params[] =
                tokenizeString(s_arr[1], ",");

                String param;
                actualParams.clear();
                for(int i =0; i < actual_params.length; i++){
                    param = actual_params[i];
                    if(param.contains("=")){
                        //If parameter specified a
                    param =
                    param.substring(param.indexOf("=")+1, param.length());
                }
                actualParams.add(param);
            }
            //Expand the macro call
            mdtPtr = Integer.parseInt(MNT.get(s_arr[0]));
            //Read macro definition starting from mdtPtr till MEND
            String macroDef;
            boolean createParamArray = true;
            String def_tokens[] = {}, paramStr = "", printStr;
            while(true){
                //First line of macro definition is name and arglist
                macroDef = MDT.get(mdtPtr);
                if(createParamArray == true){
                    createFormalParamList(macroDef);
                    createParamArray = false;
                }
                else{
                    //Tokenize line of macro definition
                    tokenizeString(macroDef, " ");
                    def_tokens = tokenizeString(macroDef, " ");
                    if (def_tokens[0].equalsIgnoreCase("MEND")) {
                        break;
                    }
                    else{
                        //Replace formal parameters with actual parameters
                        paramStr =
                        replaceFormalParams(def_tokens[1]);
                        printStr = "\t" + def_tokens[0] + " " + paramStr;
                        System.out.println(printStr);
                        out_pass2.println(printStr);
                    }
                }
                mdtPtr++;
            }
        }
        else{
            //It is a line of normal assembly code
            //Print the line as it is in the output file
            System.out.println(s);
            out_pass2.println(s);
        }
    }
    input.close();
    out_pass2.close();
}

static String replaceFormalParams(String formalParamList) {
String returnStr = "";
formalParamList = formalParamList.replace("#", "");
String param_array[] = tokenizeString(formalParamList, ",");
```

```
        String actualParam;

        for (int i = 0; i < param_array.length; i++) {
            try {
                int index = Integer.parseInt(param_array[i]);
                if (index <= actualParams.size()) {
                    actualParam = actualParams.get(index - 1);
                } else {
                    actualParam = formalParams.get(index - 1);
                }
            } catch (NumberFormatException e) {
                // If it's not a number, use the token as-is
                actualParam = param_array[i];
            }
            returnStr += actualParam + ",";
        }

        // Remove trailing comma
        if (returnStr.endsWith(",")) {
            returnStr = returnStr.substring(0, returnStr.length() - 1);
        }

        return returnStr;
    }


        static void createFormalParamList(String macroDef){
            //By processing macro call generate array of actual parameters
            String argList, arg_array[];
            String s_arr[] = tokenizeString(macroDef, " ");
            //First array element will be macro name and second will be argument list
            argList = s_arr[1];
            //Separate the arguments in the list
            arg_array = tokenizeString(argList, ",");
            String param;
            formalParams.clear();
            for(int i=0; i < arg_array.length; i++){
                param = arg_array[i];
                if(param.contains("=")){
                    //If parameter specified a default value, the value will go in the list instead of param
    name
                    param = param.substring(param.indexOf("=")+1, param.length());
                }
                formalParams.add(param);
            }
        }

        static void initializeTables() throws Exception{
            MDT = new ArrayList<String>();
            MNT = new LinkedHashMap<String,String>();
            formalParams = new ArrayList<String>();
            actualParams = new ArrayList<String>();
        //Read contents of MNT.txt and create internal data structure
        BufferedReader br;
        String s;
        br = new BufferedReader(new InputStreamReader(new
        FileInputStream("A4/MNT.txt")));
        while((s = br.readLine()) != null) {
            StringTokenizer st = new StringTokenizer(s, " ", false);
            MNT.put(st.nextToken(), st.nextToken());
        }
        br.close();

        //Read contents of MDT.txt and create internal data structure
        br = new BufferedReader(new InputStreamReader(new
        FileInputStream("A4/MDT.txt")));
        while((s = br.readLine()) != null) {
            //For each line, separate out the tokens
            String s_arr[] = tokenizeString(s, " ");
            if(s_arr.length == 0){
                continue;
            }
            int index = Integer.parseInt(s_arr[0]);

            if(s_arr.length == 2){
                MDT.add(index, s_arr[1]);
            }
            else if(s_arr.length == 3){
                MDT.add(index, s_arr[1] + " " + s_arr[2]);
            }
        }
        br.close();
    }

    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;
    }
}
```

MNT.txt

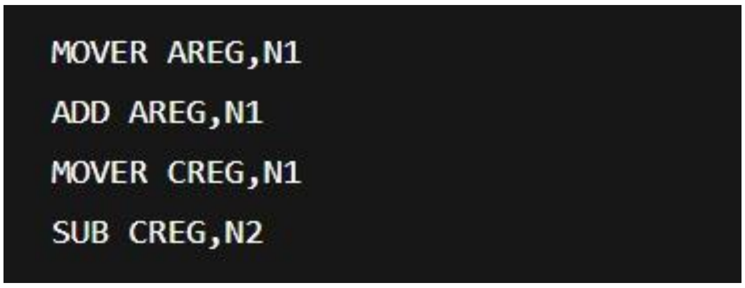
```
MAC1 0
MAC2 4
```

MDT.txt

```
 MAC1 #1,#2
 +MOVER AREG,#1
 +ADD AREG,#1
 MEND
 MAC2 #1,#2
 +MOVER CREG,#1
 +SUB CREG,#2
 MEND
```

Output_pass1.txt

```
START 100
READ N1
READ N2
MAC1 N1,N2
MAC2 N1,N2
STOP
N1 DS 1
N2 DS 1
END
```



```
MOVER AREG,N1
ADD AREG,N1
MOVER CREG,N1
SUB CREG,N2
```