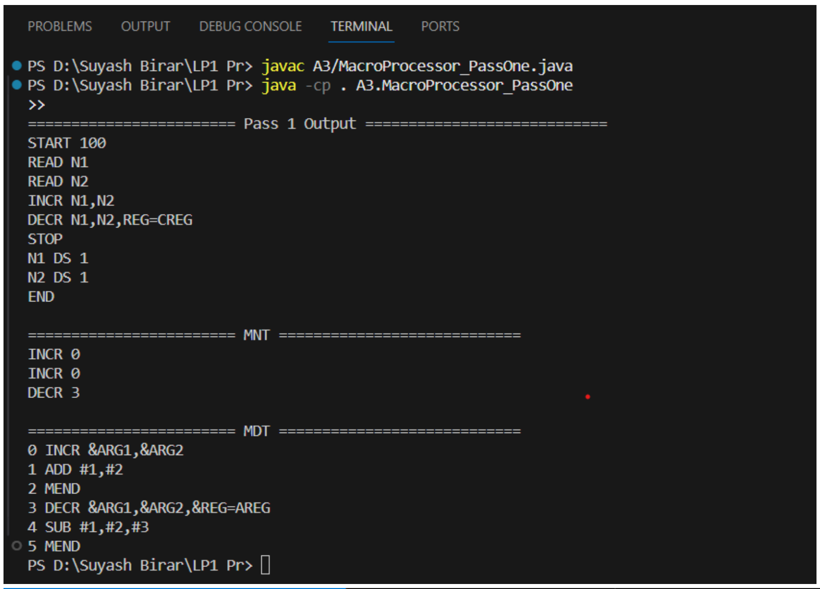
Practical No 03

```
package A3;

import java.io.*;
import java.util.*;

public class MacroProcessor_PassOne {

    static List<String> MDT = new ArrayList<>();
    static Map<String, String> MNT = new LinkedHashMap<>();
    static Map<String, String> ALA = new HashMap<>();
    static int mntPtr = 0, mdtPtr = 0;

    public static void main(String[] args) {
        try {
            pass1();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    static void pass1() throws Exception {

        BufferedReader input = new BufferedReader(new InputStreamReader(new
FileInputStream("A3/input.txt")));
        PrintWriter out_pass1 = new PrintWriter(new FileWriter("A3/output_pass1.txt"), true);
        PrintWriter out_mnt = new PrintWriter(new FileWriter("A3/MNT.txt"), true);
        PrintWriter out_mdt = new PrintWriter(new FileWriter("A3/MDT.txt"), true);

        String s;
        boolean processingMacroDefinition = false;
        boolean processMacroName = false;

        System.out.println("======================= Pass 1 Output
===========================");

        while ((s = input.readLine()) != null) {
            String[] s_arr = tokenizeString(s, " ");
            if (s_arr.length == 0) continue;

            String curToken = s_arr[0];

            if (curToken.equalsIgnoreCase("MACRO")) {
                processingMacroDefinition = true;
                processMacroName = true;
                continue;
            }

            if (processingMacroDefinition) {
                if (curToken.equalsIgnoreCase("MEND")) {
                    MDT.add(mdtPtr++, s);
                    processingMacroDefinition = false;
                    continue;
                }

                if (processMacroName) {
                    String macroName = s_arr[0];
                    String argList = s.substring(macroName.length()).trim();

                    MNT.put(macroName, String.valueOf(mdtPtr));
                    mntPtr++;

                    processMacroName = false;
                    processArgumentList(argList);
                    MDT.add(mdtPtr++, macroName + " " + argList);
                    continue;
                }

                // Convert macro body args
                String indexedLine = processArguments(s);
                MDT.add(mdtPtr++, indexedLine);

            } else {
                // Not a macro, just output it
                out_pass1.println(s);
                System.out.println(s);
            }
        }
        input.close();

        // Print MNT
        System.out.println("\n======================== MNT
============================");
        for (Map.Entry<String, String> entry : MNT.entrySet()) {
            String row = entry.getKey() + " " + entry.getValue();
            System.out.println(row);
            out_mnt.println(row);
        }

        // Print MDT
        System.out.println("\n======================== MDT
============================");
        for (int i = 0; i < MDT.size(); i++) {
            String row = i + " " + MDT.get(i);
            System.out.println(row);
            out_mdt.println(row);
        }

        out_pass1.close();
        out_mnt.close();
        out_mdt.close();
    }
```

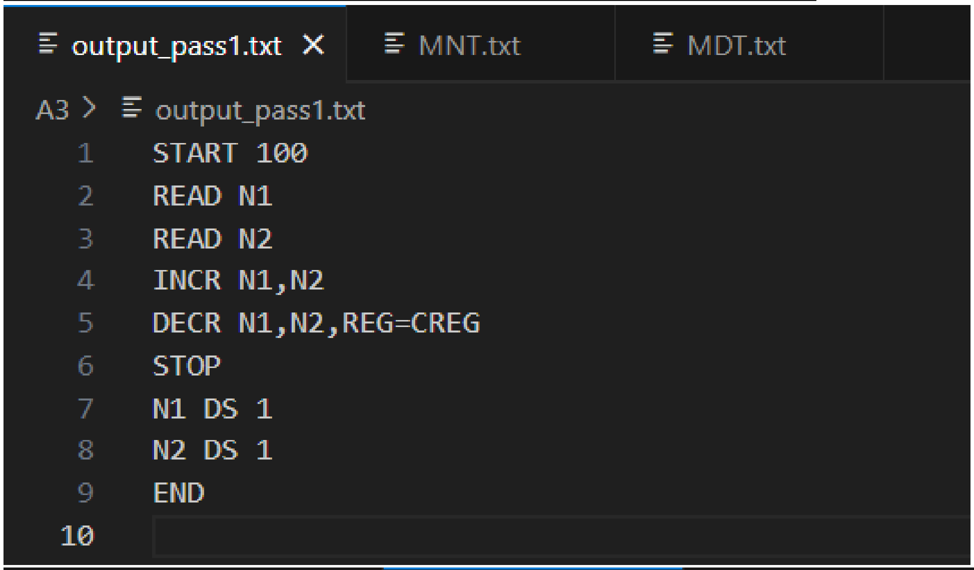
```
    static void processArgumentList(String argList) {
        StringTokenizer st = new StringTokenizer(argList, ",", false);
        ALA.clear();
        int index = 1;
        while (st.hasMoreTokens()) {
            String curArg = st.nextToken().trim();
            if (curArg.contains("=")) {
                curArg = curArg.substring(0, curArg.indexOf("="));
            }
            ALA.put(curArg, "#" + index);
            index++;
        }
    }

    static String processArguments(String line) {
        for (Map.Entry<String, String> entry : ALA.entrySet()) {
            line = line.replace(entry.getKey(), entry.getValue());
        }
        return line;
    }

    static String[] tokenizeString(String str, String separator) {
        StringTokenizer st = new StringTokenizer(str, separator, false);
        String[] s_arr = new String[st.countTokens()];
        for (int i = 0; i < s_arr.length; i++) {
            s_arr[i] = st.nextToken();
        }
        return s_arr;
    }
}
```

//input.txt

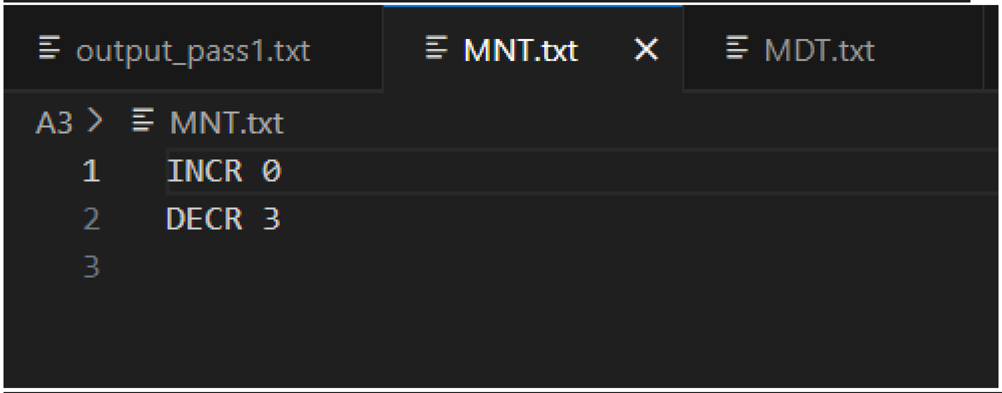
```
MACRO
INCR &ARG1,&ARG2
ADD &ARG1,&ARG2
MEND
MACRO
DECR &ARG1,&ARG2,&REG=AREG
SUB &ARG1,&ARG2,&REG
MEND
START 100
READ N1
READ N2
INCR N1,N2
DECR N1,N2,REG=CREG
STOP
N1 DS 1
N2 DS 1
END
```



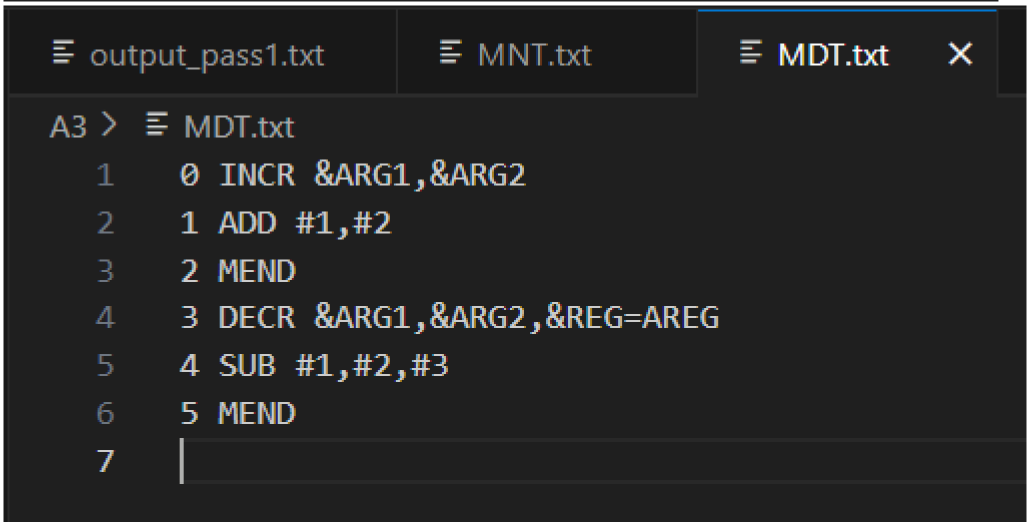
```
PS D:\Suyash Birar\LP1 Pr> javac A3/MacroProcessor_PassOne.java
PS D:\Suyash Birar\LP1 Pr> java -cp . A3.MacroProcessor_PassOne
>>
====================== Pass 1 Output ==========================
START 100
READ N1
READ N2
INCR N1,N2
DECR N1,N2,REG=CREG
STOP
N1 DS 1
N2 DS 1
END

====================== MNT ==========================
INCR 0
INCR 0
DECR 3

====================== MDT ==========================
0 INCR &ARG1,&ARG2
1 ADD #1,#2
2 MEND
3 DECR &ARG1,&ARG2,&REG=AREG
4 SUB #1,#2,#3
5 MEND
PS D:\Suyash Birar\LP1 Pr>
```



```
START 100
READ N1
READ N2
INCR N1,N2
DECR N1,N2,REG=CREG
STOP
N1 DS 1
N2 DS 1
END
```



```
INCR 0
DECR 3
```



```
0 INCR &ARG1,&ARG2
1 ADD #1,#2
2 MEND
3 DECR &ARG1,&ARG2,&REG=AREG
4 SUB #1,#2,#3
5 MEND
```