

Assignment No. 2

Name: Aryan Shinde

Div : B

Batch : C

Roll no : 58

Sub : LP-1

Pass 1 Program :

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

public class PassOne {

    static class MNTEntry {
        String name;    int
        mdtIndex;
        MNTEntry(String n, int idx) { name = n; mdtIndex = idx; }
    }

    public static void main(String[] args) throws Exception {
        String sourceFile = "input.txt";
        String mntFile = "MNT.txt";
        String mdtFile = "MDT.txt";
        String interFile = "INTERMEDIATE.txt";
        String argFile = "ARG.txt";

        List<MNTEntry> mnt = new ArrayList<>();
        List<String> mdt = new ArrayList<>();
        mdt.add(null);

        Map<String, List<String>> macroToFormals = new LinkedHashMap<>(); //

        BufferedReader br = new BufferedReader(new FileReader(sourceFile));
        BufferedWriter interBw = new BufferedWriter(new FileWriter(interFile));

        String line;
        int mdtPtr = 1;

        while ((line = br.readLine()) != null) {
            line = line.trim();
            if (line.equalsIgnoreCase("MACRO")) {
```

```

String header = br.readLine();          if (header
== null) break;          header = header.trim();
    String[] headerParts = header.split("\\s+", 2);
    String macroName = headerParts[0].trim();
    String paramPart = (headerParts.length > 1) ? headerParts[1].trim() : "";
    String[] formals = new String[0];          if
(!paramPart.isEmpty()) {
        formals = Arrays.stream(paramPart.split(","))
            .map(String::trim)
            .filter(s -> !s.isEmpty())
            .toArray(String[]::new);
    }

    mnt.add(new MNTEntry(macroName, mdtPtr));
macroToFormals.put(macroName, Arrays.asList(formals));

    Map<String, String> ala = new LinkedHashMap<>();
    for (int i = 0; i < formals.length; i++) {
        ala.put(formals[i], "#" + (i + 1));
    }
    while ((line = br.readLine()) != null) {
line = line.trim();
        if (line.equalsIgnoreCase("MEND")) {          mdt.add(mdtPtr,
"MEND");
            mdtPtr++;          break;
        } else if (line.length() == 0) {
        } else {
            String transformed = replaceFormalsWithDummies(line, ala);
mdt.add(mdtPtr, transformed);
            mdtPtr++;
        }
    }
    } else {          if
(!line.isEmpty()) {          interBw.write(line);
        interBw.newLine();
    } else {
        interBw.newLine();
    }
    }
}

br.close();
interBw.close();

```

```

        try (BufferedWriter mntBw = new BufferedWriter(new FileWriter(mntFile))) {
for (int i = 0; i < mnt.size(); i++) {      MNTEntry e = mnt.get(i);
    mntBw.write((i + 1) + "\t" + e.name + "\t" + e.mdtIndex);
    mntBw.newLine();
        }
    }

        try (BufferedWriter mdtBw = new BufferedWriter(new FileWriter(mdtFile))) {
for (int i = 1; i < mdt.size(); i++) {      mdtBw.write(i + "\t" + mdt.get(i));
    mdtBw.newLine();
        }
    }

        try (BufferedWriter argBw = new BufferedWriter(new FileWriter(argFile))) {
for (Map.Entry<String, List<String>> me : macroToFormals.entrySet()) {
    argBw.write(me.getKey() + " : ");      argBw.write(String.join(" ", me.getValue()));
    argBw.newLine();
        }
    }

    System.out.println("Pass-I complete. Files generated: " + mntFile + ", " + mdtFile + ", " +
interFile + ", " + argFile);
}

    private static String replaceFormalsWithDummies(String line, Map<String, String> ala) {
List<String> formals = new ArrayList<>(ala.keySet());    formals.sort((a,b) ->
Integer.compare(b.length(), a.length()));    String result = line;    for (String f :
formals) {        String dummy = ala.get(f);
        result = result.replace(f, dummy);
    }
    return result;
}
}

```

Text File Input :

```

MACRO
INCR &ARG1,&ARG2,&ARG3
ADD 1,&ARG1
ADD 2,&ARG2
ADD 3,&ARG3
MEND

```

```
START
LOOP INCR DATA1, DATA2, DATA3
END
```

Output :

```
PS C:\Users\durve\Desktop\MACRO> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-
XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\durve\AppData\Roaming\Code\User\workspaceStorage\78a43cc9231d57f7f398
9
8da005bf0ec\redhat.java\jdt_ws\MACRO_e0e62e1b\bin' 'PassOne'
Pass-I complete. Files generated: MNT.txt, MDT.txt, INTERMEDIATE.txt, ARG.txt
```

MNT.txt :

```
1      INCR 1
```

MDT.txt :

```
1      ADD 1,#1
2      ADD 2,#2
3      ADD 3,#3
4      MEND
```

INTERMEDIATE.txt :

```
START
LOOP INCR DATA1, DATA2, DATA3
END
```

ARG.txt :

```
INCR : &ARG1, &ARG2, &ARG3
```

PASS 2 Program

```
:
import java.io.*;
import java.util.*;

public class PassTwo {

    public static void main(String[] args) throws Exception {
        String mntFile = "MNT.txt";
        String mdtFile = "MDT.txt";
        String interFile = "INTERMEDIATE.txt";
        String outFile = "EXPANDED.txt";
```

```

Map<String, Integer> mntMap = new LinkedHashMap<>();
try (BufferedReader br = new BufferedReader(new FileReader(mntFile))) {
    String l;
    while ((l = br.readLine()) != null) {
        l = l.trim();
        if (l.isEmpty()) continue;
        String[] parts = l.split("\\s+");    if
        (parts.length >= 3) {
            String name = parts[1];        int idx
            = Integer.parseInt(parts[2]);
            mntMap.put(name, idx);
        }
    }
}
Map<Integer, String> mdt = new TreeMap<>();
try (BufferedReader br = new BufferedReader(new FileReader(mdtFile))) {
    String l;
    while ((l = br.readLine()) != null) {
l = l.trim();
        if (l.isEmpty()) continue;
        String[] parts = l.split("\\s+", 2);        int
        idx = Integer.parseInt(parts[0]);
        String body = (parts.length > 1) ? parts[1] : "";        mdt.put(idx,
body);
    }
}

    BufferedReader interBr = new BufferedReader(new FileReader(interFile));
    BufferedWriter outBw = new BufferedWriter(new FileWriter(outFile));

    String line;
    while ((line = interBr.readLine()) != null) {
line = line.trim();
        if (line.isEmpty()) {
            outBw.newLine();
            continue;
        }

        String[] parts = line.split("\\s+", 3);
        String potential1 = parts[0];
        String macroName = null;
        String label = null;
        String actualPart = "";

```

```

        if (mntMap.containsKey(potential1)) {
macroName = potential1;
            if (parts.length >= 2) actualPart = (line.substring(potential1.length())).trim(); } else
if (parts.length >= 2 && mntMap.containsKey(parts[1])) {                label = parts[0];
macroName = parts[1];                if
(parts.length == 3) actualPart = parts[2].trim();
        }

        if (macroName == null) {
outBw.write(line);
            outBw.newLine();
        } else {
            int startIndex = mntMap.get(macroName);
            String[] actuals = parseActuals(actualPart);

            Map<String, String> callALA = new HashMap<>();
if (label != null && !label.isEmpty()) {
callALA.put("#0", label);
            }
            for (int i = 0; i < actuals.length; i++) {
                callALA.put("#" + (i + 1), actuals[i]);
            }
            int ip = startIndex;
while (true) {
                String mline = mdt.get(ip);
                if (mline == null) break;
                if (mline.equalsIgnoreCase("MEND")) break;
                String expanded = replaceDummiesWithActuals(mline, callALA);
outBw.write(expanded);
                outBw.newLine();
            ip++;
        }
    }
}

interBr.close();    outBw.close();
System.out.println("Pass-II complete. Output written to: " + outFile);
}

```

```

private static String[] parseActuals(String actualPart) {
    if (actualPart == null || actualPart.trim().isEmpty()) return new String[0];

```

```

actualPart = actualPart.trim();
    String[] arr = Arrays.stream(actualPart.split(", "))
        .map(String::trim)
        .filter(s -> !s.isEmpty())
        .toArray(String[]::new);    return arr;
    }

    private static String replaceDummiesWithActuals(String line, Map<String, String> callALA)
    {
        List<String> keys = new ArrayList<>(callALA.keySet()); keys.sort((a,b)
        -> Integer.compare(b.length(), a.length())); String result = line; for
        (String k : keys) {
            result = result.replace(k, callALA.get(k));
        }
        return result;
    }
}

```

OUTPUT :

```

PS C:\Users\durve\Desktop\MACRO> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-
XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\durve\AppData\Roaming\Code\User\workspaceStorage\78a43cc9231d57f7f398
9
8da005bf0ec\redhat.java\jdt_ws\MACRO_e0e62e1b\bin' 'PassTwo'
Pass-II complete. Output written to: EXPANDED.txt

```

EXPANDED.txt :

```

START
ADD 1,DATA1
ADD 2,DATA2
ADD 3,DATA3
END

```