Student Name	**Swanand Garge**
PRN No	2280030433
Roll No	39
Program	Computer Engg.
Year	Third Year
Division	D (D2)
Subject	Systems Programming (BTECCE22504)
Assignment No	5

- **Design suitable data structures and implement simple Macro expansion for the hypothetical ALP. Generate Actual parameter table, Input file contains multiple macro calls (Minimum 3) . Assume macro definitions are stored in MDT created on Assignment 4.**

- **Output : Submit a single .doc / .pdf file containing input , MNT, MDT, APTAB and expanded code.**

INPUT of previous Assignment 4 :-

```
Macro Information Table:
Name    #PP #KP #EV MDTP    KPDTP   SSTP
INCR    2   1   1   0   0   0
DECR    2   1   1   8   1   2

MDT (Macro Definition Table):
Index   Definition
0   LCL (E,0)
1   (E,0) SET 0
2   .LOOP MOVER (P,2), (P,0)
3   ADD (P,2), ='1'
4   MOVEM (P,2), (P,0)
5   (E,0) SET (E,0)+1
6   AIF ((E,0) LT (P,1)) .LOOP
7   MEND
8   LCL (E,1)
9   (E,1) SET (P,2)
10  .START SUB (P,0), ='1'
11  (E,1) SET (E,1)-1
12  AIF ((E,1) GT (P,1)) .START
13  MEND

PNTAB for INCR Macro:
Index   Parameter
0   ARG1,
1   ARG2,
2   REG
```

MACRO CALL FILE INPUT :-

```
START

INCR NUM1, 5, AREG

DECR X, 10

MOVER AREG, NUM2

INCR NUM2, 3, BREG

DECR Y, 7, Z=5

ADD AREG, NUM1

END
```

OUTPUT :-

MNT

```
Macro Name Table (MNT):
=======================
Name    #PP #KP #EV MDTP    KPDTP   SSTP
INCR    2   1   1   0   0   0
DECR    2   1   1   8   1   2
```

MDT

```
MDT (Macro Definition Table):
Index   Definition
0   LCL (E,0)
1   (E,0) SET 0
2   .LOOP MOVER (P,2), (P,0)
3   ADD (P,2), ='1'
4   MOVEM (P,2), (P,0)
5   (E,0) SET (E,0)+1
6   AIF ((E,0) LT (P,1)) .LOOP
7   MEND
8   LCL (E,1)
9   (E,1) SET (P,2)
10  .START SUB (P,0), ='1'
11  (E,1) SET (E,1)-1
12  AIF ((E,1) GT (P,1)) .START
13  MEND
```

EXPANDED CODE :-

```
Expanded Code:
==============
START
TEMP SET 0
.LOOP MOVER PARAM2, NUM1,
ADD PARAM2, ='1'
MOVEM PARAM2, NUM1,
TEMP SET TEMP+1
AIF (TEMP LT 5,) .LOOP
VAR SET PARAM2
.START SUB X,, ='1'
VAR SET VAR-1
AIF (VAR GT 10) .START
MOVER AREG, NUM2
TEMP SET 0
.LOOP MOVER PARAM2, NUM2,
ADD PARAM2, ='1'
MOVEM PARAM2, NUM2,
TEMP SET TEMP+1
AIF (TEMP LT 3,) .LOOP
VAR SET PARAM2
.START SUB Y,, ='1'
VAR SET VAR-1
AIF (VAR GT 7,) .START
ADD AREG, NUM1
END
```

APTAB

```
APTAB (Actual Parameter Table):
==============================
Macro Call 1:
Index   Value
0   NUM1,
1   5,
2   PARAM2

Macro Call 2:
Index   Value
0   X,
1   10
2   PARAM2

Macro Call 3:
Index   Value
0   NUM2,
1   3,
2   PARAM2

Macro Call 4:
Index   Value
0   Y,
1   7,
2   PARAM2
```

CODE :-

```
import java.io.BufferedReader;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class SP_5 {
    // File names
    private static final String PREVIOUS_OUTPUT_FILE = "output.txt";
    private static final String MACRO_CALL_INPUT_FILE = "input1.txt";
    private static final String EXPANDED_OUTPUT_FILE = "expanded_output.txt";

    static class MacroInfo {
        String name;
        int pp, kp, mdtp, kpdtp, sstp, evn;
        List<String> PNTAB = new ArrayList<>();

        MacroInfo(String name, int pp, int kp, int mdtp, int kpdtp, int sstp,
int evn) {
            this.name = name;
            this.pp = pp;
            this.kp = kp;
            this.mdtp = mdtp;
            this.kpdtp = kpdtp;
            this.sstp = sstp;
            this.evn = evn;
        }
    }

    static Map<String, MacroInfo> macroInfoMap = new HashMap<>();
    static List<String> MDT = new ArrayList<>();
    static List<String> KPDTAB = new ArrayList<>();
    static List<String> EVNTAB = new ArrayList<>();
    static List<String> SSNTAB = new ArrayList<>();
    static List<Integer> SSTAB = new ArrayList<>();
    static List<List<String>> APTAB = new ArrayList<>();

    public static void main(String[] args) throws IOException {
        loadPreviousOutput(PREVIOUS_OUTPUT_FILE);
        expandMacros(MACRO_CALL_INPUT_FILE, EXPANDED_OUTPUT_FILE);
        System.out.println("Macro expansion completed. Output written to " +
EXPANDED_OUTPUT_FILE);
```

```
    }

    static void loadPreviousOutput(String filename) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(filename));
        String line;
        String currentSection = "";

        while ((line = br.readLine()) != null) {
            line = line.trim(); // Remove leading/trailing whitespace
            if (line.startsWith("Macro Information Table:")) {
                currentSection = "MNT";
                br.readLine(); // Skip header
            } else if (line.startsWith("MDT (Macro Definition Table):")) {
                currentSection = "MDT";
                br.readLine(); // Skip header
            } else if (line.startsWith("KPDTAB (Keyword Parameter Default
Table):")) {
                currentSection = "KPDTAB";
                br.readLine(); // Skip header
            } else if (line.startsWith("EVNTAB (Expansion Variable Name
Table):")) {
                currentSection = "EVNTAB";
                br.readLine(); // Skip header
            } else if (line.startsWith("SSNTAB (Sequencing Symbol Name
Table):")) {
                currentSection = "SSNTAB";
                br.readLine(); // Skip header
            } else if (line.startsWith("SSTAB (Sequencing Symbol Table):")) {
                currentSection = "SSTAB";
                br.readLine(); // Skip header
            } else if (!line.isEmpty()) {
                switch (currentSection) {
                    case "MNT":
                        String[] mntParts = line.split("\\s+");
                        if (mntParts.length >= 7) {
                            try {
                                MacroInfo info = new MacroInfo(mntParts[0],
                                    Integer.parseInt(mntParts[1]),
                                    Integer.parseInt(mntParts[2]),
                                    Integer.parseInt(mntParts[4]),
                                    Integer.parseInt(mntParts[5]),
                                    Integer.parseInt(mntParts[6]),
                                    Integer.parseInt(mntParts[3]));
                                macroInfoMap.put(mntParts[0], info);
                            } catch (NumberFormatException e) {
                                System.err.println("Error parsing MNT line: "
+ Arrays.toString(mntParts));
                            }
```

```
                        }
                        break;
                    case "MDT":
                        MDT.add(line.split("\\s+", 2)[1]);
                        break;
                    case "KPDTAB":
                        KPDTAB.add(line.split("\\s+", 2)[1]);
                        break;
                    case "EVNTAB":
                        EVNTAB.add(line.split("\\s+", 2)[1]);
                        break;
                    case "SSNTAB":
                        SSNTAB.add(line.split("\\s+", 2)[1]);
                        break;
                    case "SSTAB":
                        SSTAB.add(Integer.parseInt(line.split("\\s+", 2)[1]));
                        break;
                }
            }
        }
        br.close();

        // Load PNTAB for each macro
        for (MacroInfo info : macroInfoMap.values()) {
            String[] mdtLineParts = MDT.get(info.mdtp).split("\\s+");
            for (int i = 0; i < info.pp + info.kp; i++) {
                if (i + 1 < mdtLineParts.length) {
                    info.PNTAB.add(mdtLineParts[i + 1].substring(1));
                } else {
                    info.PNTAB.add("PARAM" + i); // Handle missing parameters
                }
            }
        }
    }

    static void expandMacros(String inputFile, String outputFile) throws
IOException {
        BufferedReader br = new BufferedReader(new FileReader(inputFile));
        PrintWriter pw = new PrintWriter(new FileWriter(outputFile));
        String line;

        pw.println("Expanded Code:");
        pw.println("==============");

        while ((line = br.readLine()) != null) {
            String[] parts = line.trim().split("\\s+");
            if (macroInfoMap.containsKey(parts[0])) {
                expandMacroCall(parts, pw);
```

```
            } else {
                pw.println(line);
            }
        }

        br.close();

        // Print APTAB in a tabular format
        pw.println("\nAPTAB (Actual Parameter Table):");
        pw.println("==============================");
        for (int i = 0; i < APTAB.size(); i++) {
            pw.println("Macro Call " + (i + 1) + ":");
            pw.println("Index\tValue");
            for (int j = 0; j < APTAB.get(i).size(); j++) {
                pw.println(j + "\t" + APTAB.get(i).get(j));
            }
            pw.println(); // Add a blank line between macro calls
        }

        pw.close();
    }

    static void expandMacroCall(String[] parts, PrintWriter pw) {
        String macroName = parts[0];
        MacroInfo info = macroInfoMap.get(macroName);
        List<String> actualParams = new
ArrayList<>(Arrays.asList(parts).subList(1, parts.length));

        List<String> aptabEntry = new ArrayList<>(info.PNTAB);
        for (int i = 0; i < actualParams.size(); i++) {
            if (i < info.pp) {
                if (i < aptabEntry.size()) {
                    aptabEntry.set(i, actualParams.get(i));
                } else {
                    aptabEntry.add(actualParams.get(i));
                }
            } else {
                String[] keywordParam = actualParams.get(i).split("=");
                int index = aptabEntry.indexOf(keywordParam[0]);
                if (index != -1) {
                    aptabEntry.set(index, keywordParam[1]);
                }
            }
        }
        APTAB.add(aptabEntry);

        for (int i = info.mdtp + 1; i < MDT.size() &&
!MDT.get(i).equals("MEND"); i++) {
```

```
            String expandedLine = expandMacroLine(MDT.get(i), APTAB.size() -
1);
            pw.println(expandedLine);
        }
    }

    static String expandMacroLine(String line, int aptabIndex) {
        for (int i = 0; i < APTAB.get(aptabIndex).size(); i++) {
            line = line.replace("(P," + i + ")",
APTAB.get(aptabIndex).get(i));
        }
        for (int i = 0; i < EVNTAB.size(); i++) {
            line = line.replace("(E," + i + ")", EVNTAB.get(i));
        }
        return line;
    }
}
```