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	4

- **Design suitable data structures and implement simple Macro definition processing for the hypothetical ALP. Generate different Parameter Tables and MDT, MNT. Detect any one error. Input file contains multiple macro definitions.**

- **Output : Submit a single .doc / .pdf file containing input ALP, MNT, MDT, PNTAB , KPDTAB , EVNTAB , SSNTAB , SSTAB in that sequence.**

ALP CODE :-

```
MACRO

INCR &ARG1, &ARG2, &REG=AREG

LCL &TEMP

&TEMP SET 0

.LOOP MOVER &REG, &ARG1

ADD &REG, ='1'

MOVEM &REG, &ARG1

&TEMP SET &TEMP+1

AIF (&TEMP LT &ARG2) .LOOP

MEND


MACRO

DECR &X, &Y, &Z=10

LCL &VAR

&VAR SET &Z

.START SUB &X, ='1'

&VAR SET &VAR-1
```

```
AIF (&VAR GT &Y) .START

MEND


START 100

INCR N, 5, REG=BREG

DECR A, 3, Z=8

END
```

OUTPUT :-

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
```

PNTABS

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

PNTAB for DECR Macro:
Index    Parameter
0        X,
1        Y,
2        Z
```

KPDTAB

```
KPDTAB (Keyword Parameter Default Table):
Index    Parameter
0        REG=AREG
1        Z=10
```

EVNTAB

```
EVNTAB (Expansion Variable Name Table):
Index    Variable
0        TEMP
1        VAR
```

SSNTAB

```
SSNTAB (Sequencing Symbol Name Table):
Index    Symbol
0        LOOP
1        IF
2        START
```

SSTAB

```
SSTAB (Sequencing Symbol Table):
Index   MDT Index
0       2
1       6
2       10
```

CODE :-

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

class SP_4 {
    static class MacroInfo {
        String name;
        int pp, kp, mdtp, kpdtp, sstp, evn;
        List<String> PNTAB = new ArrayList<>(); // Each macro has its own
PNTAB

        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<String> PNTAB = new ArrayList<>();

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader("input.txt"));
```

```
        String line;

        while ((line = br.readLine()) != null) {
            if (line.trim().startsWith("MACRO")) {
                processMacroDefinition(br);
            }
        }

        br.close();

        // Print all tables with proper labels and index numbers
        System.out.println("Macro Information Table:");
        System.out.println("Name\t#PP\t#KP\t#EV\tMDTP\tKPDTP\tSSTP");
        for (MacroInfo info : macroInfoMap.values()) {
            System.out.printf("%s\t%d\t%d\t%d\t%d\t%d\t%d\n",
                info.name, info.pp, info.kp, info.evn, info.mdtp, info.kpdtp,
info.sstp);
        }

        System.out.println("\nMDT (Macro Definition Table):");
        System.out.println("Index\tDefinition");
        for (int i = 0; i < MDT.size(); i++) {
            System.out.println(i + "\t" + MDT.get(i));
        }

        // Print separate PNTAB for each macro
        System.out.println("\nPNTAB for INCR Macro:");
        printPNTAB("INCR");

        System.out.println("\nPNTAB for DECR Macro:");
        printPNTAB("DECR");

        System.out.println("\nKPDTAB (Keyword Parameter Default Table):");
        System.out.println("Index\tParameter");
        for (int i = 0; i < KPDTAB.size(); i++) {
            System.out.println(i + "\t" + KPDTAB.get(i));
        }

        System.out.println("\nEVNTAB (Expansion Variable Name Table):");
        System.out.println("Index\tVariable");
        for (int i = 0; i < EVNTAB.size(); i++) {
            System.out.println(i + "\t" + EVNTAB.get(i));
        }

        System.out.println("\nSSNTAB (Sequencing Symbol Name Table):");
        System.out.println("Index\tSymbol");
        for (int i = 0; i < SSNTAB.size(); i++) {
            System.out.println(i + "\t" + SSNTAB.get(i));
```

```
        }

        System.out.println("\nSSTAB (Sequencing Symbol Table):");
        System.out.println("Index\tMDT Index");
        for (int i = 0; i < SSTAB.size(); i++) {
            System.out.println(i + "\t" + SSTAB.get(i));
        }
    }

    static void processMacroDefinition(BufferedReader br) throws IOException {
        String line = br.readLine();
        String[] parts = line.trim().split("\\s+");
        String macroName = parts[0];

        int pp = 0, kp = 0, evn = 0;
        List<String> macroPNTAB = new ArrayList<>(); // Local PNTAB for the
current macro
        MacroInfo currentMacro = new MacroInfo(macroName, 0, 0, MDT.size(),
KPDTAB.size(), SSTAB.size(), 0);

        // Parse parameters
        for (int i = 1; i < parts.length; i++) {
            String paramName = parts[i].substring(1);
            if (paramName.contains("=")) {
                kp++;
                KPDTAB.add(paramName);
                // Ensure the default parameter is also added to macroPNTAB if
it's not already present
                if (!macroPNTAB.contains(paramName.split("=")[0])) {
                    macroPNTAB.add(paramName.split("=")[0]);
                }
            } else {
                pp++;
                if (!macroPNTAB.contains(paramName)) {
                    macroPNTAB.add(paramName);
                }
            }
        }

        // Store the macro's PNTAB in the MacroInfo object
        currentMacro.PNTAB.addAll(macroPNTAB);

        // Process each line until "MEND"
        while (!(line = br.readLine()).trim().equals("MEND")) {
            if (line.trim().startsWith("LCL")) {
                String[] evParts = line.trim().split("\\s+");
                for (int i = 1; i < evParts.length; i++) {
                    EVNTAB.add(evParts[i].substring(1));
```

```
                    evn++;
                }
            } else if (line.contains(".")) {
                String[] ssParts = line.trim().split("\\s+");
                String ssName = ssParts[0].substring(1);
                if (!SSNTAB.contains(ssName)) {
                    SSNTAB.add(ssName);
                    SSTAB.add(MDT.size());
                }
            }

            // Replace parameters for this line and add to MDT
            String processedLine = replaceParameters(line, macroPNTAB);
            MDT.add(processedLine);
        }

        // Add "MEND" at the end
        MDT.add("MEND");

        // Update macro information
        currentMacro.pp = pp;
        currentMacro.kp = kp;
        currentMacro.evn = evn;
        macroInfoMap.put(macroName, currentMacro);
    }

    static String replaceParameters(String line, List<String> macroPNTAB) {
        // Replace positional parameters in PNTAB with (P,i)
        for (int i = 0; i < macroPNTAB.size(); i++) {
            String param = macroPNTAB.get(i).trim().replace(",", ""); //
Remove trailing comma
            String paramPlaceholder = "&" + param; // Forming the parameter
like &ARG1, &ARG2, etc.
            if (line.contains(paramPlaceholder)) {
                System.out.println("Replacing " + paramPlaceholder + " with
(P," + i + ")");
                line = line.replace(paramPlaceholder, "(P," + i + ")"); //
Replace positional parameters with (P,i)
            }
        }

        // Replace expansion variables in EVNTAB with (E,i)
        for (int i = 0; i < EVNTAB.size(); i++) {
            String evnParam = "&" + EVNTAB.get(i).trim(); // Forming the local
variables like &TEMP, &VAR, etc.
            if (line.contains(evnParam)) {
                System.out.println("Replacing " + evnParam + " with (E," + i +
")");
```

```
                line = line.replace(evnParam, "(E," + i + ")"); // Replace
expansion variables with (E,i)
            }
        }

        return line;
    }

    static void printPNTAB(String macroName) {
        MacroInfo macroInfo = macroInfoMap.get(macroName);
        if (macroInfo != null) {
            System.out.println("Index\tParameter");
            for (int i = 0; i < macroInfo.PNTAB.size(); i++) {
                System.out.println(i + "\t" + macroInfo.PNTAB.get(i));
            }
        } else {
            System.out.println("No PNTAB found for macro: " + macroName);
        }
    }
}
```