**ASSIGNMENT 4**

**SYSTEM PROGRAMMING LAB**

**202100102**

**MERWIN PINTO**

**ROLL NO 1**

**DIV E**

**MACRO INPUT  
MACRO**

**MAC1 &VAL1=, &VAL2=, &REG=AREG**

**MOVER &REG, &VAL1**

**ADD &REG, &VAL2**

**MOVEM &REG, &VAL1**

**MEND**

**CODE:**

import java.io.*\**;

import java.util.*\**;

public class MacroP1 {

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new FileReader("macro\_input.txt"));

        FileWriter macro\_name\_table = new FileWriter("macro\_name\_table.txt");

        FileWriter macro\_defination\_table = new FileWriter("macro\_defination\_table.txt");

        FileWriter keyword\_param\_default\_table = new FileWriter("keyword\_param\_default\_table.txt");

        FileWriter parameter\_name\_table = new FileWriter("parameter\_name\_table.txt");

        FileWriter evntab = new FileWriter("evntab.txt");

        FileWriter seq\_symbol\_name\_table = new FileWriter("seq\_symbol\_name\_table.txt");

        FileWriter seq\_symbol\_table = new FileWriter("seq\_symbol\_table.txt");

        LinkedHashMap<String, Integer> parameter\_name\_tableab = new LinkedHashMap<>();

        String line;

        String Macroname = null;

        int macro\_defination\_tablep = 1, keyword\_param\_default\_tablep = 0, paramNo = 1, pp = 0, kp = 0, flag = 0;

        boolean evFound = false;

        boolean labelFound = false;

        int lineIndex = 0;

        while ((line = br.readLine()) != null)

        {

            lineIndex++;

            String parts[] = line.split("\\s+");

            if (parts[0].equalsIgnoreCase("MACRO"))

            {

                flag = 1;

                line = br.readLine();

                parts = line.split("\\s+");

                Macroname = parts[0];

                if (parts.length <= 1)

                {

                    macro\_name\_table.write(parts[0] + "\t" + pp + "\t" + kp + "\t" + macro\_defination\_tablep + "\t" + (kp == 0 ? keyword\_param\_default\_tablep : (keyword\_param\_default\_tablep + 1)) + "\n");

                    continue;

                }

                for (int i = 1; i < parts.length; i++)

                {

                    parts[i] = parts[i].replaceAll("[&,]", "");

                    if (parts[i].contains("="))

                    {

                        ++kp;

                        String keywordParam[] = parts[i].split("=");

                        parameter\_name\_tableab.put(keywordParam[0], paramNo++);

                        if (keywordParam.length == 2)

                        {

                            keyword\_param\_default\_table.write(keywordParam[0] + "\t" + keywordParam[1] + "\n");

                        }

                        else

                        {

                            keyword\_param\_default\_table.write(keywordParam[0] + "\t-\n");

                        }

                    }

                    else {

                        parameter\_name\_tableab.put(parts[i], paramNo++);

                        pp++;

                    }

                }

                macro\_name\_table.write("MN" + "\t \t" + "PP" + "\t" + "KP" + "\t" + "MDT" + "\t" + "KPDT" +"\n");

                macro\_name\_table.write(parts[0] + "\t" + pp + "\t" + kp + "\t" + macro\_defination\_tablep + "\t" + (kp == 0 ? keyword\_param\_default\_tablep : (keyword\_param\_default\_tablep + 1)) + "\n");

                keyword\_param\_default\_tablep = keyword\_param\_default\_tablep + kp;

            }

            else if (parts[0].equalsIgnoreCase("MEND"))

            {

                macro\_defination\_table.write(line + "\n");

                flag = kp = pp = 0;

                macro\_defination\_tablep++;

                paramNo = 1;

                parameter\_name\_table.write(Macroname + ":\t");

                Iterator<String> itr = parameter\_name\_tableab.keySet().iterator();

                while (itr.hasNext())

                {

                    parameter\_name\_table.write(itr.next() + "\t");

                }

                parameter\_name\_table.write("\n");

                parameter\_name\_tableab.clear();

            }

            else if (flag == 1)

            {

                for (int i = 0; i < parts.length; i++)

                {

                    if (parts[i].contains("&"))

                    {

                        parts[i] = parts[i].replaceAll("[&,]", "");

                        macro\_defination\_table.write("(P," + parameter\_name\_tableab.get(parts[i]) + ")\t");

                    }

                    else

                    {

                        macro\_defination\_table.write(parts[i] + "\t");

                    }

                }

                macro\_defination\_table.write("\n");

                macro\_defination\_tablep++;

            }

            else

            {

            }

            if (line.contains("LCL"))

            {

                String[] p = line.split("\\s+");

                for (String part : p)

                {

                    if (part.equals("LCL"))

                    {

                        continue;

                    }

                    String cleanPart = part.replace("&", "");

                    evntab.write(cleanPart + "\n");

                    evFound = true;

                }

            }

            for (int i = 0; i < parts.length; i++)

            {

                String part = parts[i].trim();

                if (part.startsWith(".") && part.length() > 1) {

                    seq\_symbol\_name\_table.write(part + "\n");

                    seq\_symbol\_table.write(lineIndex + 1 + "\n");

                    labelFound = true;

                }

            }

        }

        if (!evFound) {

            evntab.write("No EV found\n");

        }

        if (!labelFound) {

            seq\_symbol\_name\_table.write("No sequencing symbols found\n");

        }

        br.close();

        macro\_defination\_table.close();

        macro\_name\_table.close();

        parameter\_name\_table.close();

        keyword\_param\_default\_table.close();

        evntab.close();

        seq\_symbol\_name\_table.close();

        seq\_symbol\_table.close();

        System.out.println("Macro Pass1 Processing done.");

    }

}

**OUTPUT**

**EV TAB**

**NO EV**

**KPDT**

**VAL1 -**

**VAL2 -**

**REG AREG**

**MDT**

**MOVER (P,3) (P,1)**

**ADD (P,3) (P,2)**

**MOVEM (P,3) (P,1)**

**MEND**

**MNT**

**MN PP KP MDT KPDT**

**MAC1 0 3 1 1**

**PNTAB**

**MAC1: VAL1 VAL2 REG   
  
SS**

**No sequencing symbols found**