Practical No. 2

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

public class TwoPassMacroProcessor {

    static class MNTEntry {
        String name;
        int mdtIndex;
        int paramCount;

        MNTEntry(String name, int mdtIndex, int paramCount) {
            this.name = name;
            this.mdtIndex = mdtIndex;
            this.paramCount = paramCount;
        }
    }

    static List<String> MDT = new ArrayList<>();
    static Map<String, MNTEntry> MNT = new HashMap<>();

    static final String MACRO = "MACRO";
    static final String MEND = "MEND";

    public static void main(String[] args) throws IOException {
        String inputFile = "input.asm";
        String intermediateFile = "intermediate.asm";
        String outputFile = "output.asm";

        passOne(inputFile, intermediateFile);
        passTwo(intermediateFile, outputFile);
    }

    static void passOne(String inputFile, String intermediateFile) throws IOException {
        BufferedReader br = new BufferedReader(new FileReader(inputFile));
        BufferedWriter bw = new BufferedWriter(new FileWriter(intermediateFile));

        String line;
        boolean inMacroDef = false;
        String macroName = null;

        while ((line = br.readLine()) != null) {
            line = line.trim();
            if (line.isEmpty()) continue;

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

            if (tokens[0].equalsIgnoreCase(MACRO)) {

                line = br.readLine();
                if (line == null) break;
```

```
            line = line.trim();

            String[] headerTokens = line.split("\\s+", 2);
            macroName = headerTokens[0];
            String[] params = new String[0];
            if (headerTokens.length > 1) {
                params = headerTokens[1].split(",");
            }
            inMacroDef = true;

            MNT.put(macroName, new MNTEntry(macroName, MDT.size(), params.length));

            MDT.add(line);
            continue;
        }

        if (inMacroDef) {
            if (tokens[0].equalsIgnoreCase(MEND)) {
                MDT.add(MEND);
                inMacroDef = false;
                continue;
            } else {
                MNTEntry entry = MNT.get(macroName);
                String defLine = line;

                String headerLine = MDT.get(entry.mdtIndex);
                String[] headerParts = headerLine.split("\\s+", 2);

                if (headerParts.length > 1) {
                    String[] paramNames = headerParts[1].split(",");
                    for (int p = 0; p < paramNames.length; p++) {
                        String pname = paramNames[p].trim();
                        if (!pname.isEmpty()) {

                            defLine = defLine.replaceAll("\\b" + pname + "\\b", "#" + (p + 1));
                        }
                    }
                }
                MDT.add(defLine);
            }
        } else {

            bw.write(line);
            bw.newLine();
        }
    }
    br.close();
    bw.close();
}

static void passTwo(String intermediateFile, String outputFile) throws IOException {
    BufferedReader br = new BufferedReader(new FileReader(intermediateFile));
```

```
        BufferedWriter bw = new BufferedWriter(new FileWriter(outputFile));

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

            String[] tokens = line.split("\\s+");
            String firstToken = tokens[0];

            if (MNT.containsKey(firstToken)) {
                MNTEntry entry = MNT.get(firstToken);


                String argsPart = line.substring(firstToken.length()).trim();
                String[] args = argsPart.isEmpty() ? new String[0] : argsPart.split("\\s*,\\s*");


                int mdtIndex = entry.mdtIndex + 1;
                while (mdtIndex < MDT.size() && !MDT.get(mdtIndex).equalsIgnoreCase(MEND)) {
                    String mdtLine = MDT.get(mdtIndex);


                    for (int i = 0; i < args.length; i++) {
                        mdtLine = mdtLine.replaceAll("\\#" + (i + 1), args[i].trim());
                    }

                    bw.write(mdtLine);
                    bw.newLine();

                    mdtIndex++;
                }
            } else {

                bw.write(line);
                bw.newLine();
            }
        }
        br.close();
        bw.close();
    }
}
```

input.asm:

```
MACRO
INCR &A
LOAD &A
ADD ONE
STORE &A
MEND

START
INCR COUNT
END
```

intermediate.asm:

```
START
INCR COUNT
END
```

output.asm:

```
START
LOAD &A
ADD ONE
STORE &A
END
```