Lab Assignment No. 5

Pass-I and Pass-II of a two-pass macroprocessor

Program:

```
import java.util.*;

public class TwoPassMacroProcessor {

    // Macro Name Table Entry
    static class MNTEntry {
        String name;
        int mdtIndex;

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

    public static void main(String[] args) {
        // Sample input program with macro definitions
        String[] input = {
            "MACRO",
            "INCR &ARG1,&ARG2",
            "LDA &ARG1",
            "ADD &ARG2",
            "STA &ARG1",
            "MEND",
            "START",
            "INCR A,B",
            "END"
        };

        List<MNTEntry> MNT = new ArrayList<>();
        List<String> MDT = new ArrayList<>();
        List<String> intermediateCode = new ArrayList<>();

        pass1(input, MNT, MDT, intermediateCode);
        List<String> expandedCode = pass2(MNT, MDT, intermediateCode);

        System.out.println("MNT:");
        for (MNTEntry e : MNT) {
            System.out.println(e.name + " -> MDT index: " + e.mdtIndex);
        }
        System.out.println("\nMDT:");
        for (int i = 0; i < MDT.size(); i++) {
            System.out.println(i + ": " + MDT.get(i));
        }
        System.out.println("\nIntermediate Code:");
        for (String line : intermediateCode) {
            System.out.println(line);
        }
        System.out.println("\nExpanded Code:");
        for (String line : expandedCode) {
            System.out.println(line);
        }
    }
    static void pass1(String[] input, List<MNTEntry> MNT, List<String> MDT, List<String>
intermediateCode) {
        boolean insideMacro = false;
        String currentMacro = "";

        for (String line : input) {
            line = line.trim();
            if (line.equals("")) continue;

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

            if (tokens[0].equals("MACRO")) {
                insideMacro = true;
                continue;
            }

            if (insideMacro) {
                if (tokens[0].equals("MEND")) {
                    MDT.add("MEND");
                    insideMacro = false;
                    currentMacro = "";
                    continue;
                }

                if (currentMacro.equals("")) {
                    // Macro prototype line: e.g. INCR &ARG1,&ARG2
                    currentMacro = tokens[0];
                    MNT.add(new MNTEntry(currentMacro, MDT.size()));
                    MDT.add(line);
                } else {
                    MDT.add(line);
                }
            } else {
                intermediateCode.add(line);
            }
        }
    }

    static List<String> pass2(List<MNTEntry> MNT, List<String> MDT, List<String>
intermediateCode) {
        List<String> expandedCode = new ArrayList<>();

        for (String line : intermediateCode) {
            String[] tokens = line.split("\\s+");
            if (tokens.length == 0) continue;

            // Check if this line is a macro call
            MNTEntry macroEntry = null;
            for (MNTEntry e : MNT) {
                if (tokens[0].equals(e.name)) {
                    macroEntry = e;
                    break;
                }
            }

            if (macroEntry != null) {
                // Macro call detected
```

```
int mdtIndex = macroEntry.mdtIndex;
String prototypeLine = MDT.get(mdtIndex);
// Parse formal args from prototype line
String[] protoTokens = prototypeLine.split("\\s+");
String formalArgStr = protoTokens.length > 1 ? protoTokens[1] : "";
String[] formalArgs = formalArgStr.split(",");

// Parse actual args from call line
String actualArgStr = tokens.length > 1 ? tokens[1] : "";
String[] actualArgs = actualArgStr.split(",");

// Build argument map (ALA)
Map<String, String> ALA = new HashMap<>();
for (int i = 0; i < formalArgs.length; i++) {
    String formal = formalArgs[i].trim();
    String actual = i < actualArgs.length ? actualArgs[i].trim() : "";
    ALA.put(formal, actual);
}

// Expand macro lines
int i = mdtIndex + 1;
while (!MDT.get(i).equals("MEND")) {
    String expandedLine = MDT.get(i);
    for (Map.Entry<String, String> entry : ALA.entrySet()) {
        expandedLine = expandedLine.replace(entry.getKey(), entry.getValue());
    }
    expandedCode.add(expandedLine);
    i++;
```

```
                }
            } else {
                // Normal line, just copy
                expandedCode.add(line);
            }
        }

        return expandedCode;
    }
}
```

Output:

```
PS C:\Users\CC\Desktop\Assembler>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-
XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\CC\AppData\Roaming\Code\User\workspaceStorage\fa003457982375af06d05f2a33f7
1c87\redhat.java\jdt_ws\Assembler_fff334fd\bin' 'TwoPassMacroProcessor'

MNT:
INCR -> MDT index: 0

MDT:
0: INCR &ARG1,&ARG2
1: LDA &ARG1
2: ADD &ARG2
3: STA &ARG1
4: MEND

Intermediate Code:
START
INCR A,B
END
```

```
Expanded Code:
START
LDA A
ADD B
STA A
END
PS C:\Users\CC\Desktop\Assembler>
```