MPass2:

```
import java.io.*;
import java.util.*;
class MntEntry {
   String name;
   int mdtAddress;
   int argCount;
   public MntEntry(String name, int mdtAddress, int argCount) {
      this.name = name;
      this.mdtAddress = mdtAddress;
      this.argCount = argCount;
   }
}
public class MPass2 {
   public static void main(String[] args) throws IOException {
      List<MntEntry> mnt = new ArrayList<>();
      List<String> mdt = new ArrayList<>();
      BufferedReader mntReader = new BufferedReader(new FileReader("MNT.txt"));
      String line;
      while ((line = mntReader.readLine()) != null) {
         String[] parts = line.split("\\s+");
         mnt.add(new MntEntry(parts[0], Integer.parseInt(parts[1]),
Integer.parseInt(parts[2])));
      }
      mntReader.close();
      BufferedReader mdtReader = new BufferedReader(new FileReader("MDT.txt"));
      while ((line = mdtReader.readLine()) != null) {
```

```
            mdt.add(line);
        }
        mdtReader.close();
        BufferedReader inputReader = new BufferedReader(new
FileReader("input_pass2.txt"));
        BufferedWriter outputWriter = new BufferedWriter(new FileWriter("output.txt"));
        System.out.println("\n---------- EXPANDED CODE (output.txt) ----------");
        while ((line = inputReader.readLine()) != null) {
            line = line.trim().replaceAll(",", "");
            String[] tokens = line.split("\\s+");
            String instruction = tokens[0];
            MntEntry calledMacro = null;
            for (MntEntry entry : mnt) {
                if (entry.name.equalsIgnoreCase(instruction)) {
                    calledMacro = entry;
                    break;
                }
            }
            if (calledMacro != null) {
                int actualArgCount = tokens.length - 1;
                if (actualArgCount != calledMacro.argCount) {
                    System.err.println("ERROR: Argument count mismatch for macro " +
calledMacro.name);
                    outputWriter.write("; ERROR: Argument count mismatch for macro " +
calledMacro.name + "\n");
                    continue; // Skip expansion
                }
                Map<Integer, String> actualArgMap = new HashMap<>();
                for (int i = 1; i < tokens.length; i++) {
```

```
            actualArgMap.put(i, tokens[i]);
        }
        int mdtIndex = calledMacro.mdtAddress + 1; // Start from line after macro definition
        while (!mdt.get(mdtIndex).equalsIgnoreCase("MEND")) {
            String mdtLine = mdt.get(mdtIndex);
            String[] mdtTokens = mdtLine.split("\\s+");
            StringBuilder expandedLine = new StringBuilder();
            expandedLine.append(mdtTokens[0]); // Append instruction
            for (int i = 1; i < mdtTokens.length; i++) {
                if (mdtTokens[i].startsWith("#")) {
                    int argIndex = Integer.parseInt(mdtTokens[i].substring(1));
                    expandedLine.append("\t").append(actualArgMap.get(argIndex));
                } else {
                    expandedLine.append("\t").append(mdtTokens[i]);
                }
            }
            outputWriter.write(expandedLine.toString() + "\n");
            System.out.println(expandedLine.toString());
            mdtIndex++;
        }
    } else {
        // Not a macro call, write the line as is
        outputWriter.write(line + "\n");
        System.out.println(line);
    }
}
```

```
        inputReader.close();
        outputWriter.close();
        System.out.println("\n-----------------------------------------------");
        System.out.println("Pass 2 processing complete. Check output.txt.");
    }
}
```

Output: