import java.io.\*;

import java.util.\*;

class SolutionPass2 {

    BufferedReader input; // Input from pass-1 output

    BufferedReader MNT;

    BufferedReader MDT;

    BufferedWriter output; // Final output file after macro expansion

    Map<String, Integer> macroTable = new HashMap<>(); // Macro name and MDT pointer

    List<String> MDTContent = new ArrayList<>(); // Stores all MDT lines

    public SolutionPass2() throws IOException {

        input = new BufferedReader(new FileReader("D:\\java\\output.asm")); // Pass-1 output file

        MNT = new BufferedReader(new FileReader("D:\\java\\MNT.file"));

        MDT = new BufferedReader(new FileReader("D:\\java\\MDT.file"));

        output = new BufferedWriter(new FileWriter("D:\\java\\expanded\_output.asm"));

        loadMNT(); // Load the Macro Name Table

        loadMDT(); // Load the Macro Definition Table

    }

    // Load the Macro Name Table (MNT) from the MNT file

    public void loadMNT() throws IOException {

        String line;

        while ((line = MNT.readLine()) != null) {

            // Split by whitespace

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

            // Ensure the line has a valid format (macroName mdtPointer)

            if (parts.length < 2) {

                System.out.println("Malformed MNT entry: " + line);

                continue;

            }

            macroTable.put(parts[0], Integer.parseInt(parts[1])); // Macro name and MDT pointer

        }

    }

    // Load the entire MDT into a list (MDTContent)

    public void loadMDT() throws IOException {

        String line;

        while ((line = MDT.readLine()) != null) {

            MDTContent.add(line); // Add all lines of MDT into list

        }

    }

    public void expandMacros() throws IOException {

        String card = input.readLine();

        while (card != null) {

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

            // If this is a macro invocation

            if (macroTable.containsKey(parts[0])) {

                int mdtIndex = macroTable.get(parts[0]);

                List<String> ala = new ArrayList<>();

                // Read the actual arguments from macro invocation

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

                    ala.add(parts[i]);

                }

                // Expand the macro body from MDT

                expandMDT(mdtIndex, ala);

            } else {

                // If not a macro, write the assembly code directly

                output.write(card + "\n");

                output.flush();

            }

            card = input.readLine();

        }

        // Close resources

        output.close();

        MNT.close();

        MDT.close();

        input.close();

    }

    // Expand the macro using MDT and ALA (Argument List Array)

    // Expand the macro using MDT and ALA (Argument List Array)

    public void expandMDT(int mdtIndex, List<String> ala) throws IOException {

        String mdtLine;

        while (!(mdtLine = MDTContent.get(mdtIndex)).equals("MEND")) {

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

            StringBuilder expandedLine = new StringBuilder();

            // Replace parameters with actual arguments from ALA

            for (String token : tokens) {

                if (token.startsWith("&")) {

                    int paramIndex = ala.indexOf(token); // Locate the token in ala

                    if (paramIndex != -1) {

                        expandedLine.append(ala.get(paramIndex)).append(" ");

                    } else {

                        System.out.println("Error processing macro parameter: " + token);

                    }

                } else {

                    expandedLine.append(token).append(" ");

                }

            }

            output.write(expandedLine.toString().trim() + "\n");

            output.flush();

            mdtIndex++;

        }

    }

}

public class Pass\_2 {

    public static void main(String[] args) {

        try {

            SolutionPass2 s = new SolutionPass2();

            s.expandMacros();

        } catch (IOException e) {

            System.out.println("File error: " + e.getMessage());

        }

    }

}