import java.io.\*;

import java.util.HashMap;

import java.util.Map;

class Solution {

    BufferedReader input;

    BufferedWriter MNT;

    BufferedWriter MDT;

    BufferedWriter output;

    BufferedWriter temp;

    Map<String, Integer> macroTable = new HashMap<>();

    int mntc;

    int mdtp;

    public Solution() throws IOException {

        input = new BufferedReader(new FileReader("D:\\java\\input.asm"));

        MNT = new BufferedWriter(new FileWriter("D:\\java\\mnt.file"));

        MDT = new BufferedWriter(new FileWriter("D:\\java\\mdt.file"));

        output = new BufferedWriter(new FileWriter("D:\\java\\output.asm"));  // Output file

        mntc = 0;

        mdtp = 0;

    }

    public void generateOutput() throws IOException {

        String card = input.readLine();

        boolean flag = false; // Denotes whether we are in a MACRO or out

        // Handle macro definitions

        while (card != null && !card.contains("START")) {

            if (!card.equals("MACRO")) {

                card = input.readLine();

                continue;

            }

            flag = true; // Inside a MACRO function

            card = input.readLine();

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

            temp = new BufferedWriter(new FileWriter("D:\\java\\ALA\_" + parts[0]));

            MNT.write(parts[0] + "\t" + mdtp + "\n");

            MNT.flush();

            macroTable.put(parts[0], mdtp);  // Save macro name and MDT pointer in a map

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

                if (parts[i].contains("=")) {  // the argument with the  default value

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

                    temp.write(s[0]);

                    temp.flush();

                    continue;

                }

                temp.write(parts[i] + "\n");  // argument without default value

                temp.flush();

            }

            // Write the macro body to MDT

            while (flag) {

                card = input.readLine();

                mdtp++;

                if (card.equals("MEND")) {

                    MDT.write("MEND\n");

                    MDT.flush();

                    flag = false;

                    break;

                }

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

                MDT.flush();

            }

            card = input.readLine();

        }

        // Now handle non-macro part and macro invocations after START

        while (card != null) {

            if (macroTable.containsKey(card.split("\\s+")[0])) {  // If it's a macro invocation

                // Write macro invocation (without expanding it)

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

                output.flush();

            } else {

                // Write regular assembly code directly to output file

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

                output.flush();

            }

            card = input.readLine();

        }

        // Close resources

        output.close();

        MNT.close();

        MDT.close();

        input.close();

    }

}

public class Pass\_1 {

    public static void main(String[] args) {

        try {

            Solution s = new Solution();

            s.generateOutput();

        } catch (IOException e) {

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

        }

    }

}