import java.io.\*;

import java.util.\*;

public class Assembler1 {

private static Map<String, Integer> symbolTable = new HashMap<>();

private static List<String> litTab = new ArrayList<>();

private static List<Integer> poolTab = new ArrayList<>();

private static List<String> icList = new ArrayList<>();

private static final Map<String, Integer> optab = new HashMap<>();

public static void main(String[] args) {

initializeOptab();

try {

BufferedReader reader = new BufferedReader(new FileReader("output.txt"));

parseSymbolTable(reader);

parseLiteralTable(reader);

parsePoolTable(reader);

parseIntermediateCode(reader);

reader.close();

processIC();

} catch (IOException e) {

e.printStackTrace();

}

}

private static void initializeOptab() {

optab.put("ADD", 1);

optab.put("SUB", 2);

optab.put("MOVER", 1);

optab.put("MOVEM", 1);

optab.put("PRINT", 1);

optab.put("STOP", 1);

}

private static void parseSymbolTable(BufferedReader reader) throws IOException {

String line;

while ((line = reader.readLine()) != null && !line.equals("Literal Table:")) {

if (line.trim().isEmpty() || line.contains("Table:")) continue;

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

if (parts.length == 2) {

try {

symbolTable.put(parts[0], Integer.parseInt(parts[1]));

} catch (NumberFormatException e) {

System.out.println("Skipping invalid line: " + line);

}

}

}

}

private static void parseLiteralTable(BufferedReader reader) throws IOException {

String line;

while ((line = reader.readLine()) != null && !line.equals("Pool Table:")) {

if (line.trim().isEmpty() || line.contains("Table:")) continue;

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

if (parts.length == 2) {

litTab.add(parts[1]);

}

}

}

private static void parsePoolTable(BufferedReader reader) throws IOException {

String line;

while ((line = reader.readLine()) != null && !line.equals("Intermediate Code:")) {

if (line.trim().isEmpty() || line.contains("Table:")) continue;

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

if (parts.length == 2) {

try {

poolTab.add(Integer.parseInt(parts[1]));

} catch (NumberFormatException e) {

System.out.println("Skipping invalid line: " + line);

}

}

}

}

private static void parseIntermediateCode(BufferedReader reader) throws IOException {

String line;

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

if (line.trim().isEmpty()) continue;

icList.add(line);

}

}

private static void processIC() throws IOException {

try (PrintWriter writer = new PrintWriter(new FileWriter("machine\_code.txt"))) {

for (String ic : icList) {

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

String instruction = parts[0];

if (instruction.equals("IS")) {

String opcode = parts[1];

String operands = String.join(" ", Arrays.copyOfRange(parts, 2, parts.length));

writer.print(optab.get(opcode) + " ");

String[] operandsArr = operands.split(",\\s\*");

for (String operand : operandsArr) {

if (operand.startsWith("=")) {

int litIndex = litTab.indexOf(operand);

if (litIndex != -1) {

writer.print(litIndex + " ");

} else {

writer.print("-1 ");

}

} else if (operand.contains("+")) {

String symbol = operand.split("\\+")[0];

Integer address = symbolTable.get(symbol);

if (address != null) {

writer.print(address + " ");

} else {

writer.print("-1 ");

}

} else {

Integer address = symbolTable.get(operand);

if (address != null) {

writer.print(address + " ");

} else {

writer.print("-1 ");

}

}

}

writer.println();

} else if (instruction.equals("STOP")) {

writer.println("STOP");

} else if (instruction.equals("END")) {

writer.println("END");

}

}

}

}

}