**CODE**

import java.util.\*;

class CodeGen {

public static void main(String args[]) {

System.out.println("Enter the input expression:");

String s = new Scanner(System.in).nextLine();

char variable = 't';

char final\_variable = s.charAt(0);

Map<String, String> expressions = new TreeMap();

int open, close;

String line = new String();

do {

open = s.indexOf('(');

if(open != -1) {

close = s.indexOf(')');

String x = s.substring(open+1, close);

line += s.substring(0, open);

if(expressions.get(x) == null) {

line += variable;

expressions.put(x, Character.toString(variable++));

} else {

line += expressions.get(x);

}

s = s.substring(close+1, s.length());

}

} while(open != -1);

line = line.substring(line.indexOf('=')+1, line.length()).trim();

while(line.length() > 3) {

String expression = line.substring(0,3);

CharSequence cs1 = expression;

String replacement = expressions.get(expression);

CharSequence cs2 = replacement;

if(cs2 == null) {

cs2 = Character.toString(variable);

expressions.put(expression, Character.toString(variable++));

}

line = line.replace(cs1, cs2);

}

expressions.put(line, Character.toString(final\_variable));

Iterator<String> iterator = expressions.keySet().iterator();

while(iterator.hasNext()) {

String x = iterator.next();

System.out.println(expressions.get(x) + "=" + x);

}

createTable(expressions);

}

static void createTable(Map<String, String> m) {

List<String> code = new LinkedList();

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

Set<String> expressions = m.keySet();

Iterator<String> iterator = expressions.iterator();

while(iterator.hasNext()) {

String x = iterator.next();

String var1 = "" + x.charAt(0);

String operator = "" + x.charAt(1);

String var2 = "" + x.charAt(2);

String s1 = null;

String s2 = null;

Iterator<Integer> setIterator = register.keySet().iterator();

while(setIterator.hasNext()) {

int i = setIterator.next();

String y = register.get(i);

if(y.equals(var1)) {

s1 = "" + i;

}

if(y.equals(var2)) {

s2 = "" + i;

}

}

if((s1 == null) && (s2 == null)) {

s1 = "" + register.size();

register.put(Integer.valueOf(s1), var1);

code.add("MOV " + var1 + ", R" + s1);

code.add(stringOf(operator) + " " + var2 + ", R" + s1);

register.put(Integer.valueOf(s1), m.get(x));

} else if((s1 != null) && (s2 != null)) {

code.add(stringOf(operator) + " R" + s2 + ", R" + s1);

register.put(Integer.valueOf(s1), m.get(x));

} else {

if(s1 != null) {

code.add(stringOf(operator) + " " + var2 + ", R" + s1);

register.put(Integer.valueOf(s1), m.get(x));

}

if(s2 != null) {

code.add(stringOf(operator) + " " + var1 + ", R" + s2);

register.put(Integer.valueOf(s2), m.get(x));

}

}

if(!iterator.hasNext()) {

setIterator = register.keySet().iterator();

while(setIterator.hasNext()) {

int i = setIterator.next();

String y = register.get(i);

if(y.equals(m.get(x))) {

s1 = "" + i;

code.add("MOV R" + i + ", " + m.get(x));

}

}

}

System.out.println("--------------------");

System.out.println("Statement:\n" + m.get(x) + "=" + x + "\n\nCode Generated:");

for(String i : code) {

System.out.println(i);

}

code.clear();

System.out.println("\nRegisters:\n" + register);

}

}

static String stringOf(String operator) {

if(operator.equals("+")) {

return "ADD";

} else if(operator.equals("-")) {

return "SUB";

} else if(operator.equals("\*")) {

return "MUL";

} else {

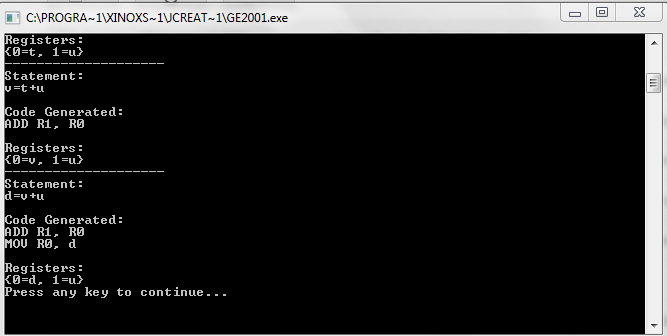
return "DIV";

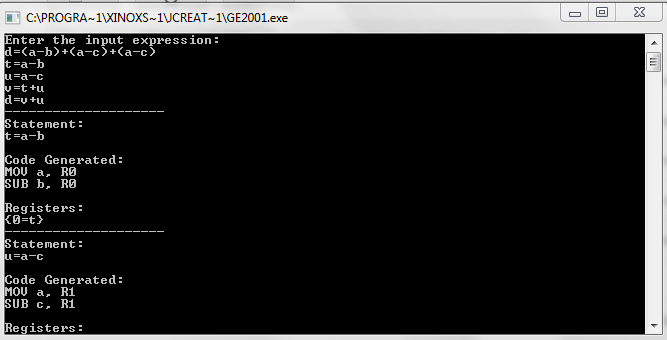
}

}

}

**OUTPUT:**

****

****