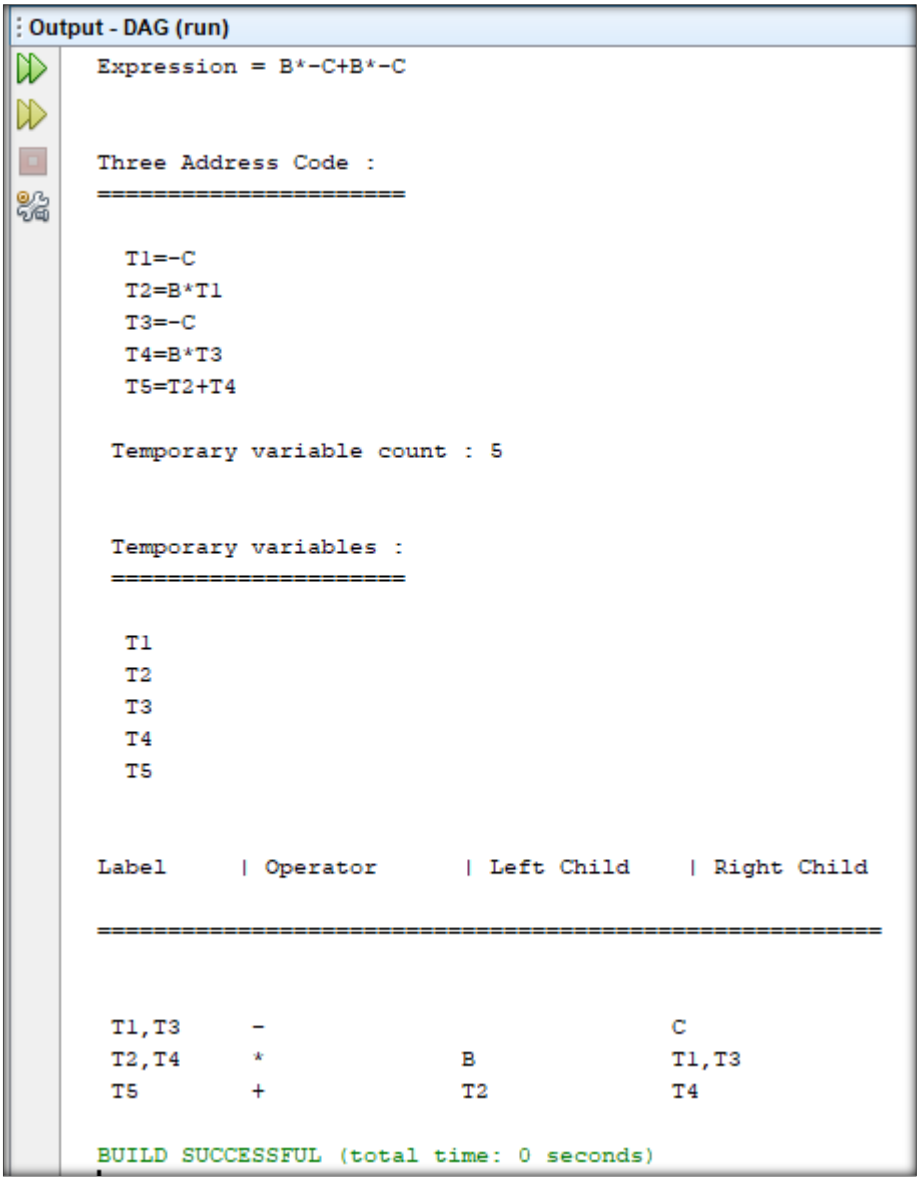
**Practical No 6**

**AIM: Write a code to generate the DAG for the input arithmetic expression.**

```
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class DAG {

   private String expression = "B*-C+B*-C";
   private List<String> tacList = new ArrayList<String>();

   public void printExpression() {
      System.out.println("\nInput Expression :");
      System.out.println("==================\n");
      System.out.println("Expression = " + expression + "\n\n");
   }

   public void convertToTAC() {
      int cnt = 1;
      String tempStr = new String();
      tempStr = "T" + cnt + "=" + expression.substring(2, 4);
      cnt++;
      tacList.add(tempStr);
      tempStr = "T" + cnt + "=" + expression.substring(0, 2) + "T" + (cnt - 1);
      cnt++;
      tacList.add(tempStr);
      tempStr = "T" + cnt + "=" + expression.substring(7, 9);
      cnt++;
      tacList.add(tempStr);
      tempStr = "T" + cnt + "=" + expression.substring(5, 7) + "T" + (cnt - 1);
      cnt++;
      tacList.add(tempStr);
      tempStr = "T5=T2+T4";
      tacList.add(tempStr);
      System.out.println("Three Address Code : ");
      System.out.println("======================\n");
      for (int i = 0; i < tacList.size(); i++) {
         System.out.println("  " + tacList.get(i));
      }
      System.out.println();
      System.out.println(" Temporary variable count : " + tacList.size() + "\n\n");
      System.out.println(" Temporary variables :");
      System.out.println(" ====================\n");
      for (int i = 0; i < tacList.size(); i++) {
         System.out.println("  T" + (i + 1));
      }
   }

   public void convertToDAG() {
      Map<String, String[]> dagMap = new LinkedHashMap<String, String[]>();
      String result;
      String[] data;
```

```
String tempStr;
String oper;
String[] tempDAGStr = new String[3];
for (int i = 0; i < tacList.size(); i++) {
   String[] DAGtable = new String[3];
   result = tacList.get(i).substring(0, tacList.get(i).indexOf("="));
   tempStr = tacList.get(i).substring(tacList.get(i).indexOf("=") + 1, tacList.get(i).length());
   data = tempStr.split("\\-|\\+|\\*|\\/|\\[\\]|\\<|\\>");

   Pattern pattern = Pattern.compile("\\-|\\+|\\*|\\/|\\[\\]|\\<|\\>");
   Matcher matcher = pattern.matcher(tempStr);
   matcher.find();
   oper = tempStr.substring(matcher.start(), matcher.start() + 1);
   if (dagMap.containsKey(oper)) {
      tempDAGStr = dagMap.get(oper);
      if (data[0].length() == 0) {
         if (tempDAGStr[2] != null && !tempDAGStr[2].contains(data[1])) {
            tempDAGStr[2] += "," + data[1];
         } else {
            tempDAGStr[2] = data[1];
         }
         if (tempDAGStr[0] != null && !tempDAGStr[0].contains(data[1])) {
            tempDAGStr[0] += "," + result;
         } else {
            tempDAGStr[0] = result;
         }
      } else {
         if (tempDAGStr[2] != null && !tempDAGStr[2].contains(data[1])) {
            tempDAGStr[2] += "," + data[1];
         } else {
            tempDAGStr[2] = data[1];
         }
         if (tempDAGStr[0] != null && !tempDAGStr[0].contains(result)) {
            tempDAGStr[0] += "," + result;
         } else {
            tempDAGStr[0] = result;
         }
         if (tempDAGStr[1] != null && !tempDAGStr[1].contains(data[0])) {
            tempDAGStr[1] += "," + data[0];
         } else {
            tempDAGStr[1] = data[0];
         }
      }
      dagMap.put(oper, tempDAGStr);
   } else {
      if (data[0].length() == 0) {
         DAGtable[2] = data[1];
         DAGtable[0] = result;
         DAGtable[1] = "";
      } else {
         DAGtable[2] = data[1];
         DAGtable[0] = result;
         DAGtable[1] = data[0];
      }
```

```
            dagMap.put(oper, DAGtable);
          }
        }
        System.out.println("\n");
        System.out.println("Label     | Operator      | Left Child    | Right Child  ");
        System.out.println("\n=======================================================\n");
        Iterator it = dagMap.entrySet().iterator();
        while (it.hasNext()) {
          Map.Entry entry = (Map.Entry) it.next();
          tempDAGStr = (String[]) entry.getValue();
          System.out.println();
          System.out.format(" %-10s%-15s%-15s%-15s", tempDAGStr[0], (String) entry.getKey(), tempDAGStr[1],
tempDAGStr[2]);
        }
        System.out.println("\n");
    }

    public static void main(String[] args) {
        DAG dag = new DAG();
        dag.printExpression();
        dag.convertToTAC();
        dag.convertToDAG();

    }

}
```


Output - DAG (run)
Expression = B*-C+B*-C
Three Address Code :
T1=-C
T2=B*T1
T3=-C
T4=B*T3
T5=T2+T4
Temporary variable count : 5
Temporary variables :
T1
T2
T3
T4
T5
Label | Operator | Left Child | Right Child
T1,T3 - C
T2,T4 * B T1,T3
T5 + T2 T4
BUILD SUCCESSFUL (total time: 0 seconds)