DESIGN AND IMPLEMENTATION OF MODERN COMPILERS

INDEX

Practical 1

Write a program to convert the given NDFA to DFA.

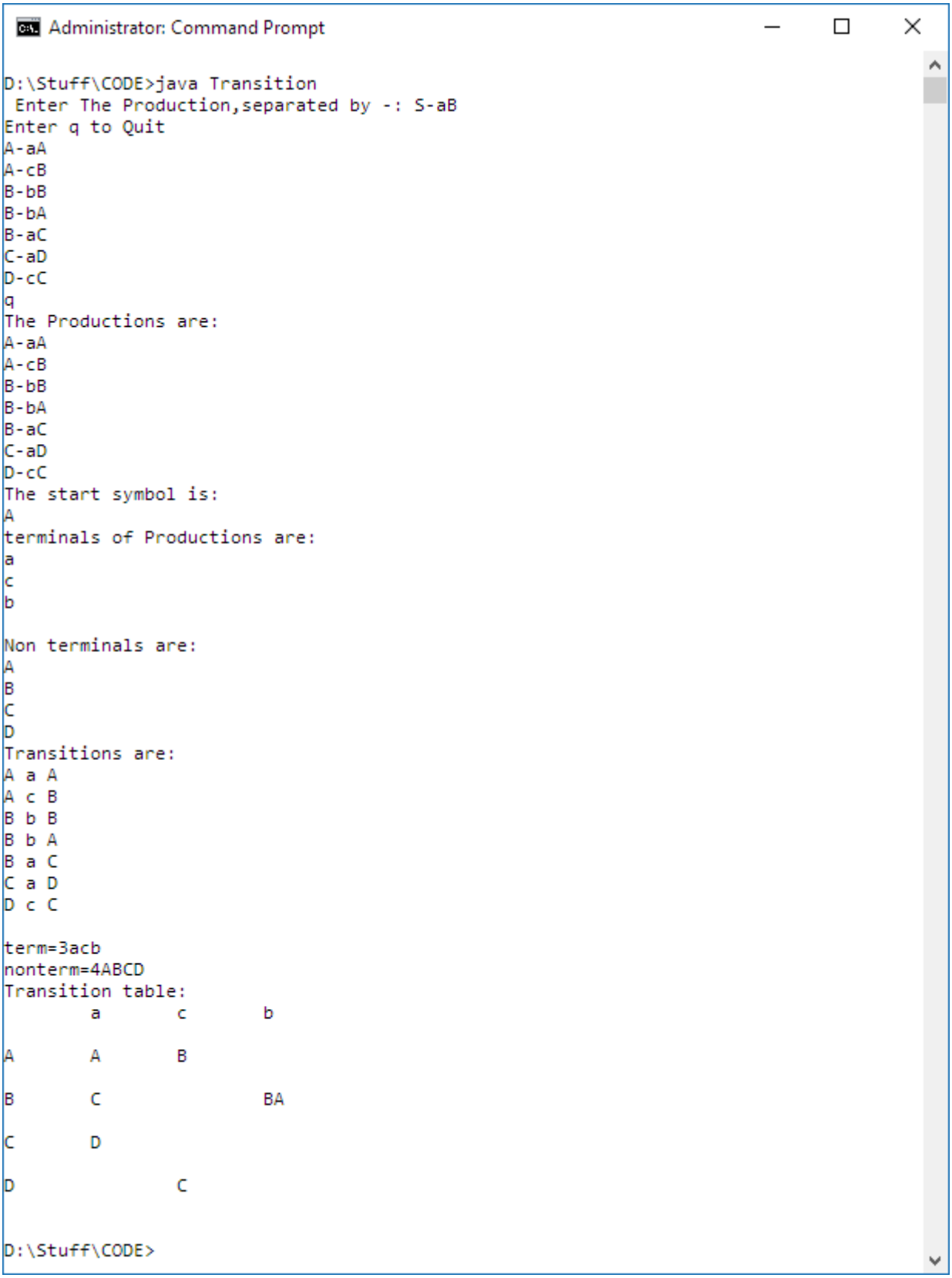
Code:

```
import java.io.*;
public class Transition {
 int i, j, k;
 char temp[] = new char[15];
 char tempn[] = new char[15];
 int cn = 0, cnt = 0;
 int ct = 0, ctt = 0;
 String str[] = new String[10];
 String transt[][];
 String tempt = "", tempnt = "";
 public void record() {
  try {
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println(" Enter The Production,separated by -: S-aB");
   System.out.println("Enter q to Quit");
   for (i = 0; i < str.length; i++) {
    str[i] = br.readLine();
    if (str[i].equals("q"))
     break;}
   System.out.println("The Productions are:");
   for (i = 0; i < str.length; i++) {
    if (str[i].equals("q"))
     break;
    System.out.println(str[i]);}}catch (IOException e) {}
  System.out.println("The start symbol is:");
  System.out.println(str[0].charAt(0));
  System.out.println("terminals of Productions are:");
  try {
   for (i = 0; i < str.length; i++) {
    if (!str[i].equals("q")) {
     if (str[i].charAt(2) >= 97 && str[i].charAt(2) <= 122) {
      temp[ct] = str[i].charAt(2);
      ct++;}}}}catch (Exception e) {}
  for (i = 0; i < ct; i++) {
   for (j = i + 1; j < ct; j++) {
    if (temp[i] == temp[j]) {
     temp[j] = '-';}}}
  for (i = 0; i < ct; i++) {
   if (temp[i] != '\0' && temp[i] != '-')
    System.out.println(temp[i]);}
  System.out.println();
```

```
System.out.println("Non terminals are:");
try {
 for (i = 0; i < str.length; i++) {
  if (!str[i].equals("q")) {
   for (j = 0; j < str[i].length(); j++) {
    if (str[i].charAt(j) >= 65 && str[i].charAt(j) <= 90) {
     tempn[cn] = str[i].charAt(j);
     cn++;}}}}}catch (Exception e) {}
for (i = 0; i < cn; i++) {
 for (j = i + 1; j < cn; j++) {
  if (tempn[i] == tempn[j]) {
   tempn[j] = '-';}}}
for (i = 0; i < cn; i++) {
 if (tempn[i] != '\0' && tempn[i] != '-')
  System.out.println(tempn[i]);}
System.out.println("Transitions are:");
try {
 for (i = 0; i < str.length; i++) {
  for (j = 0; j < str[i].length(); j++) {
   if (str[i].charAt(j) != '-' && str[i].charAt(j) != 'q')
    System.out.print(str[i].charAt(j) + " ");}
  System.out.println();}}catch (Exception e) {}}
public void ndfa() {
 int cndf;
 transt = new String[cn][ct];
 //put all the non terminals into rows and terminals into columns.
 for (i = 0; i <= ct; i++) {
  if (temp[i] != '\0' && temp[i] != '-') {
   tempt += temp[i];
   ctt++;}}
 System.out.println("term=" + ctt + tempt);
 for (i = 0; i <= cn; i++) {
  if (tempn[i] != '\0' && tempn[i] != '-') {
   tempnt += tempn[i];
   cnt++;}}
 System.out.println("nonterm=" + cnt + tempnt);
 for (i = 0; i < cn; i++) {
  for (j = 0; j < ct; j++) {
   transt[i][j] = "";}}
 for (j = 0; j < Math.max(ctt, cnt); j++) {
  if (j + 1 <= ctt)
   transt[0][j + 1] = tempt.substring(j, j + 1);
  else
   transt[0][j + 1] = "";
  if (j + 1 <= cnt)
   transt[j + 1][0] = tempnt.substring(j, j + 1);
  else
   transt[j + 1][0] = "";}
 //checking each string with each row,each row with each column.
 try {
  for (i = 0; i < str.length; i++) {
```

```
  for (j = 0; j < cnt; j++) {
   for (k = 0; k < ctt; k++) {
    if (str[i].substring(0, 1).equals(transt[j + 1][0]) &&
     str[i].substring(2, 3).equals(transt[0][k + 1])) {
     transt[j + 1][k + 1] += str[i].substring(3, 4);
    } else      transt[j + 1][k + 1] += "";}}}} catch (NullPointerException e) {}
 System.out.println("Transition table:"); //displaying Transition
 for (i = 0; i <= cnt; i++) {
  for (j = 0; j <= ctt; j++) {
   System.out.print(transt[i][j] + "\t");
  }
  System.out.println("\n");}}
public static void main(String arr[]) {
 Transition t = new Transition();  t.record();  t.ndfa();}}
```

OUTPUT


Administrator: Command Prompt
D:\Stuff\CODE>java Transition
Enter The Production,separated by -: S-aB
Enter q to Quit
A-aA
A-cB
B-bB
B-bA
B-aC
C-aD
D-cC
q
The Productions are:
A-aA
A-cB
B-bB
B-bA
B-aC
C-aD
D-cC
The start symbol is:
A
terminals of Productions are:
a
c
b
Non terminals are:
A
B
C
D
Transitions are:
A a A
A c B
B b B
B b A
B a C
C a D
D c C
term=3acb
nonterm=4ABCD
Transition table:
a c b
A A B
B C BA
C D
D C
D:\Stuff\CODE>

Practical 2

Write a program to convert the given Right Linear Grammar to Left Linear Grammar form.

Code:

```
import java.io.*;
public class Grammar {
 public static void main(String arr[]) throws IOException {
  int i, j, k;
  char temp[] = new char[20];
  char left[] = new char[20];
  char tempn[] = new char[20];
  char templt[] = new char[20];
  char templnt[] = new char[20];
  char right;
  int count = 0;  int cn = 0;  int clt = 0;  int clnt = 0;
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  String str[] = new String[20];
  System.out.println(" Enter The Right linear grammar is,separated by-: S-aB");
  System.out.println("Enter q to Quit");
  for (i = 0; i < str.length; i++) {
   str[i] = br.readLine();
   if (str[i].equals("q"))
    break;  }
  System.out.println("The Right linear grammar is:");
  for (i = 0; i < str.length; i++) {
   if (str[i].equals("q"))
    break;
   System.out.println(str[i]);
  }
  System.out.println("terminals are:");
  try {
   for (i = 0; i < str.length; i++) {
    for (j = 2; j < str[i].length(); j++) {
     if (str[i].charAt(j) >= 97 && str[i].charAt(j) <= 122) {
      if (i == 0)
       temp[count] = str[i].charAt(j);
      else {
       for (k = 0; k <= count; k++) {
        if (str[i].charAt(j) == temp[k]) {
         break;       }      }
       if (k > count) {
        count++;
        temp[count] = str[i].charAt(j);}}}}}
  catch (NullPointerException e) {}
  for (i = 0; i <= count; i++)
   System.out.print(temp[i] + " ");
  System.out.println("\nNon terminals are:");
  count = 0;
  try {
   for (i = 0; i < str.length; i++) {
```

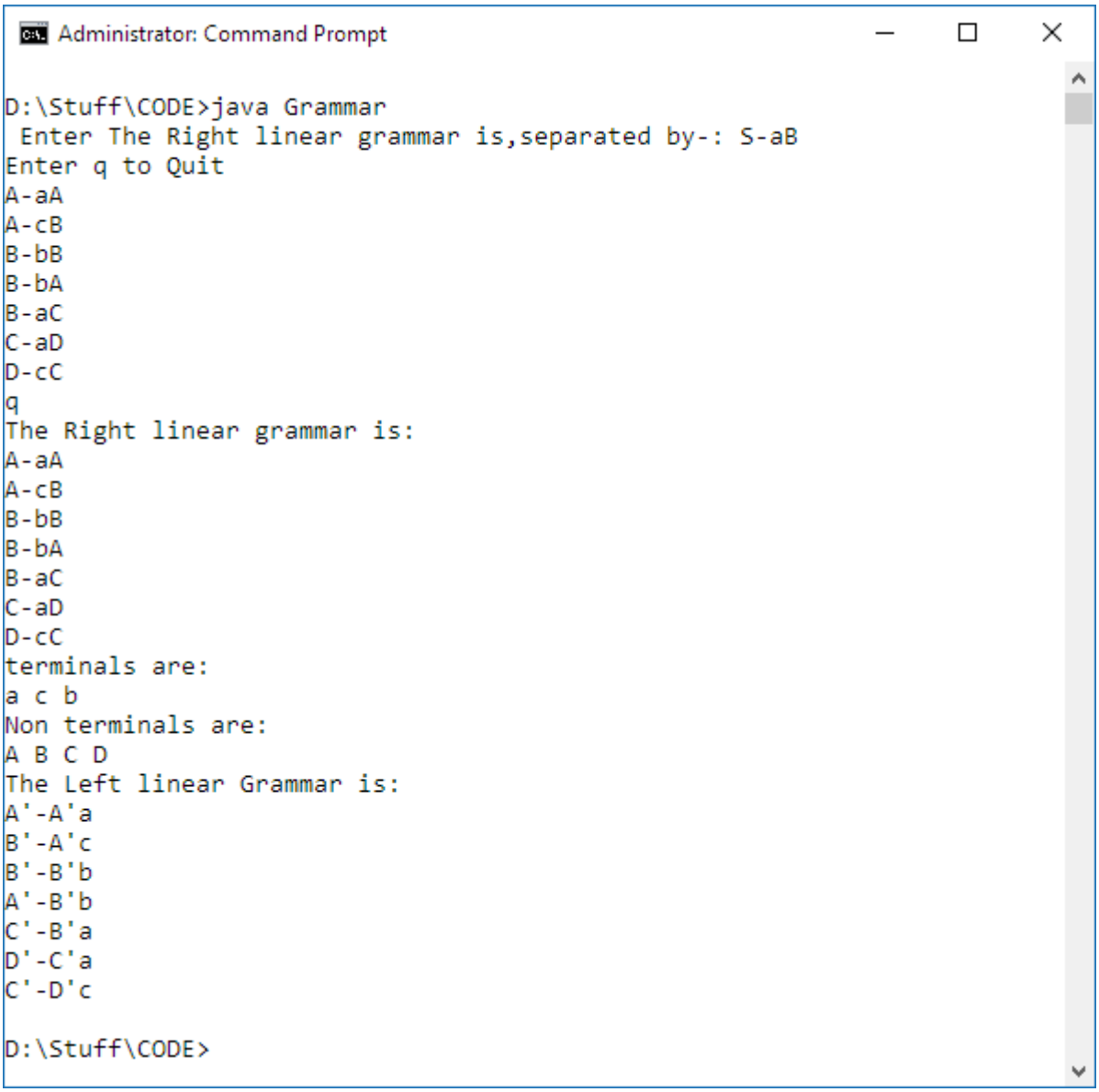
```
 for (j = 2; j < str[i].length(); j++) {
  if (str[i].charAt(j) >= 65 && str[i].charAt(j) <= 90) {
   if (i == 0)
    tempn[count] = str[i].charAt(j);
   else {

    for (k = 0; k <= count; k++) {
     if (str[i].charAt(j) == tempn[k]) {
      break;}}
    if (k > count) {
     count++;
     tempn[count] = str[i].charAt(j);}}}}} catch (NullPointerException e) {}
for (i = 0; i <= count; i++) {
 System.out.print(tempn[i] + " ");
}
System.out.println();
System.out.println("The Left linear Grammar is:");
for (i = 0; i < str.length; i++) {
 int cr = 0;
 if (str[i].equals("q"))
  break;
 //checking whether 1st & last element of productions r nonterminal.
 if ((str[i].charAt(0) >= 65 &&
   str[i].charAt(0) <= 90) && (str[i].charAt(str[i].length() - 1) >= 65 &&
   str[i].charAt(str[i].length() - 1) <= 90)) {
  if ((str[i].length() - 2) / 2 >= 1) //finding the length of terminals.
  {
   for (j = 2; j < str[i].length(); j++) {
    //Storing the terminals @ nonterminal after the terminals.
    left[cr] = str[i].charAt(j);
    cr++;
   }
   //reversing the element of left[k].
   for (k = 0; k < (str[i].length() - 2) / 2; k++) {

    //System.out.print(left[k]);
    right = left[k];
    left[k] = left[((str[i].length() - 2) - 1) - k];
    left[((str[i].length() - 2) - 1) - k] = right;
   }
   //displaying the left linear grammar in proper manner.
   System.out.print(left[0] + "\'" + "-" + str[i].charAt(0) + "\'");
   for (k = ((str[i].length() - 2) - 1); k >= 1; k--) {
    System.out.print(left[k]);
   }
   System.out.println();
  } else
   System.out.println(str[i].charAt(str[i].length() - 1) + "\'" + "-" +
    str[i].charAt(0) + "\'");
 } else {
  System.out.print("Z" + "\'" + "-" + str[i].charAt(0) + "\'");
```

```
for (j = 2; j < str[i].length(); j++)
 System.out.print(str[i].charAt(j));
System.out.println();}}}}
```

OUTPUT



Practical 3
Write a program to illustrate the generation on SPM for the input grammar.

Code:

```
import java.io.*;
public class SPM {
 int i, j, k, l;
 int count = 0;
 int ct = 0;
 String prod[] = new String[50];
 char temp[] = new char[50];
 String tempt = "";
 String str[] = new String[20];
 String firstm[][] = new String[50][50];
 String lastm[][] = new String[50][50];
 String firstp[][] = new String[50][50];
 String lastp[][] = new String[50][50];
 String lasts[][] = new String[50][50];
 String lastt[][] = new String[50][50];
 String firsts[][] = new String[50][50];
 String equalm[][] = new String[50][50];
 String less[][] = new String[50][50];
 String greater[][] = new String[50][50];
 String simplepm[][] = new String[50][50];
 public void getproduction() {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  try {
   System.out.println("Enter The Production");
   System.out.println("Enter q to quit");
   for (i = 0; i < prod.length; i++) {
    prod[i] = br.readLine();
    if (prod[i].equals("q"))
     break;
   }
   System.out.println("Produstions are:");
   for (i = 0; i < prod.length; i++) {
    if (prod[i].equals("q"))
     break;
    System.out.println(prod[i]);
   }
  } catch (IOException e) {}
  //listing set of all the terminals & nonterminals.
  try {
   for (i = 0; i < prod.length; i++) {
    for (j = 0; j < prod[i].length(); j++) {
     if (prod[i].charAt(j) != '-' && prod[i].charAt(j) != 'q') {
      temp[count] = prod[i].charAt(j);
      count++;}}}} catch (NullPointerException e) {}
  //Removing the repeatation
  for (i = 0; i < temp.length; i++) {
```

```
 for (j = i + 1; j < temp.length; j++) {
  if (temp[i] == temp[j]) {
   temp[i] = '-';}}}
 for (i = 0; i < temp.length; i++) {
  if (temp[i] != '\0' && temp[i] != '-') {
   tempt += temp[i];
   ct++;}}
 System.out.println(tempt);
}
public void first() {

 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   firstm[i][j] = "";
   lastm[i][j] = "";}}
 for (i = 0; i < ct; i++) {
  firstm[i + 1][0] += tempt.substring(i, i + 1);
 }
 for (j = 0; j < ct; j++) {
  firstm[0][j + 1] += tempt.substring(j, j + 1);
 }
 try {
  for (i = 0; i < prod.length; i++) {
   for (j = 0; j <= ct; j++) {
    for (k = 0; k <= ct; k++) {
     if (prod[i].substring(0, 1).equals(firstm[j + 1][0]) &&
      prod[i].substring(2, 3).equals(firstm[0][k + 1])) {
      firstm[j + 1][k + 1] += "1";}}}}} catch (NullPointerException e) {}
 System.out.println("First matrix----------------------------------------------------------------------");
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (firstm[i + 1][j + 1] == "")
    firstm[i + 1][j + 1] += "0";
   System.out.print(firstm[i][j] + "\t");
  }
  System.out.println("\n");}}
public void last() {
 for (i = 0; i < ct; i++) {
  lastm[i + 1][0] += tempt.substring(i, i + 1);
 }
 for (j = 0; j < ct; j++) {
  lastm[0][j + 1] += tempt.substring(j, j + 1);
 }
 try {
  for (i = 0; i < prod.length; i++) {
   for (j = 0; j <= ct; j++) {
    for (k = 0; k <= ct; k++) {
     if (prod[i].substring(0, 1).equals(lastm[j + 1][0]) &&
      prod[i].substring(prod[i].length() - 1).equals(lastm[0][k + 1])) {
      lastm[j + 1][k + 1] += "1";}}}}} catch (NullPointerException e) {}
 System.out.println("Last matrix-----------------------------------------------------------------");
```

```
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (lastm[i + 1][j + 1] == "")

    lastm[i + 1][j + 1] += "0";
   System.out.print(lastm[i][j] + "\t");
  }
  System.out.println("\n");}}
 public void firstplus() {
  int temp;
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    firstp[i][j] = firstm[i][j];}}
  for (i = 1; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    if (firstp[j][i].equals("1")) {
     for (k = 1; k <= ct; k++) {
      temp = Integer.parseInt(firstp[j][k]) | Integer.parseInt(firstp[i][k]);
      if (temp == 1)
       firstp[j][k] = "1";}}}}
  System.out.println("First+ matrix----------------------------------------------------------------");
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    if (firstp[i + 1][j + 1] == "")
     firstp[i + 1][j + 1] += "0";
    System.out.print(firstp[i][j] + "\t");
   }
   System.out.println("\n");}}

 public void lastplus() {
  int temp;
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    lastp[i][j] = lastm[i][j];}}
  for (i = 1; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    if (lastp[j][i].equals("1")) {
     for (k = 1; k <= ct; k++) {
      temp = Integer.parseInt(lastp[j][k]) | Integer.parseInt(lastp[i][k]);
      if (temp == 1)
       lastp[j][k] = "1";}}}}
  System.out.println("Last+ matrix----------------------------------------------------------------");
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    if (lastp[i + 1][j + 1] == "")
     lastp[i + 1][j + 1] += "0";
    System.out.print(lastp[i][j] + "\t");}
   System.out.println("\n");}}
 public void firststar() {
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
```

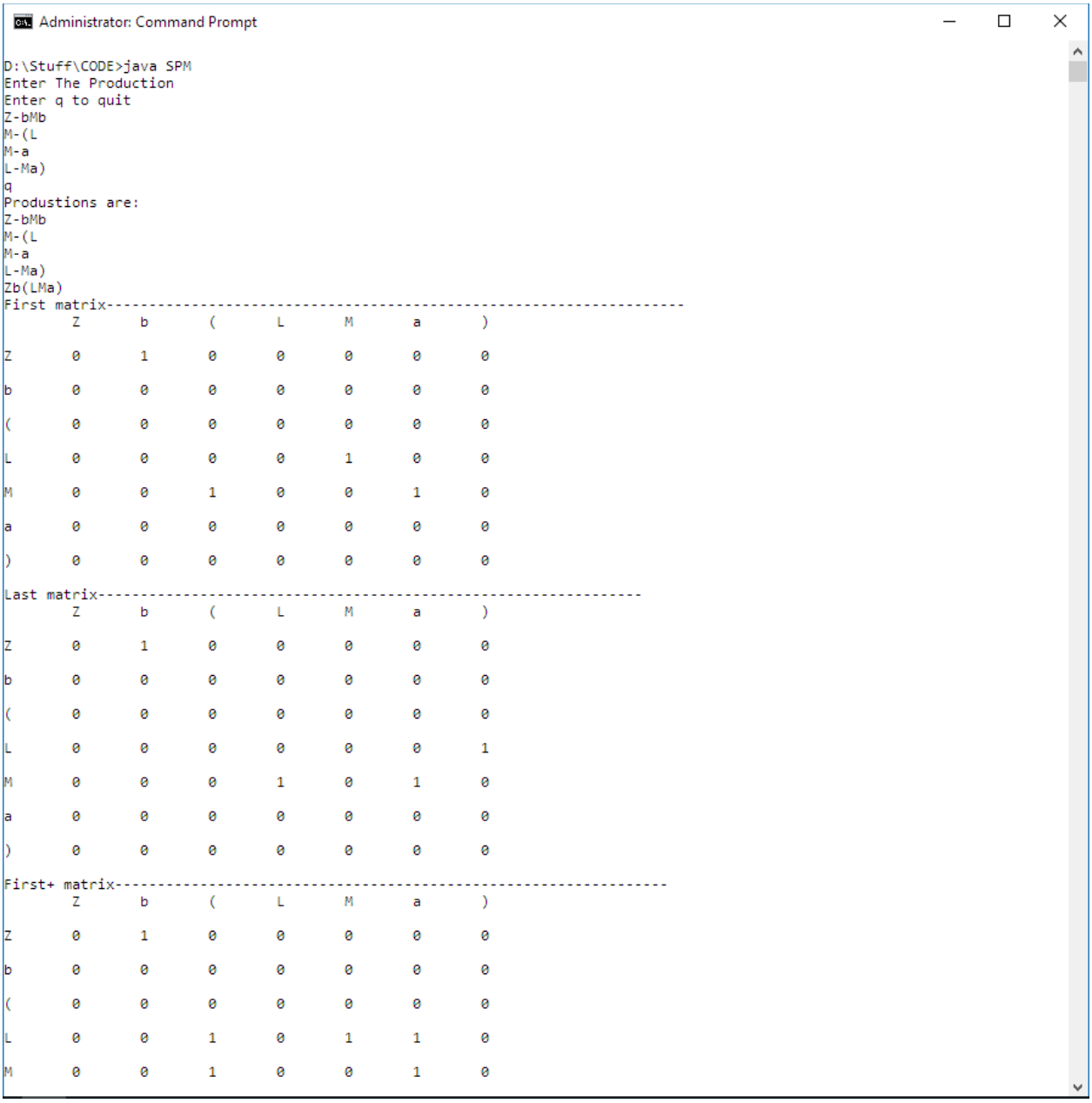
```
  if (j == i)
   firsts[i][j] = "";
  else
   firsts[i][j] = firstp[i][j];}}
 for (i = 1; i <= ct; i++) {
 j = i;
 firsts[i + 1][j + 1] += "1";}
 System.out.println("First* Matrix----------------------------------------------------------------");
 for (i = 0; i <= ct; i++) {
 for (j = 0; j <= ct; j++) {
  if (firsts[i][j].equals(""))
   firsts[i + 1][j + 1] += "0";
  System.out.print(firsts[i][j] + "\t");
 }
 System.out.println("\n");}}
public void laststar() {
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (j == i)
    lasts[i][j] = "";
   else
    lasts[i][j] = lastp[i][j];}}
 for (i = 1; i <= ct; i++) {
 j = i;
 lasts[i][j] += "1";
 }
 System.out.println("Last* Matrix----------------------------------------------------------------");
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (lasts[i][j].equals(""))
    lasts[i + 1][j + 1] += "0";
   System.out.print(lasts[i][j] + "\t");
  }
  System.out.println("\n");}}
public void equal() {
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   equalm[i][j] = "";}}
 for (i = 0; i < ct; i++) {
  equalm[i + 1][0] += tempt.substring(i, i + 1);
 }
 for (j = 0; j < ct; j++) {
  equalm[0][j + 1] += tempt.substring(j, j + 1);
 }
 try {
  for (i = 0; i < prod.length; i++) {
   try {
    if ((prod[i].substring(2).length()) >= 2); {
     for (j = 0; j <= ct; j++) {
      for (k = 0; k <= ct; k++) {
       for (l = 2; l <= prod[i].length() - 2; l++) {
```

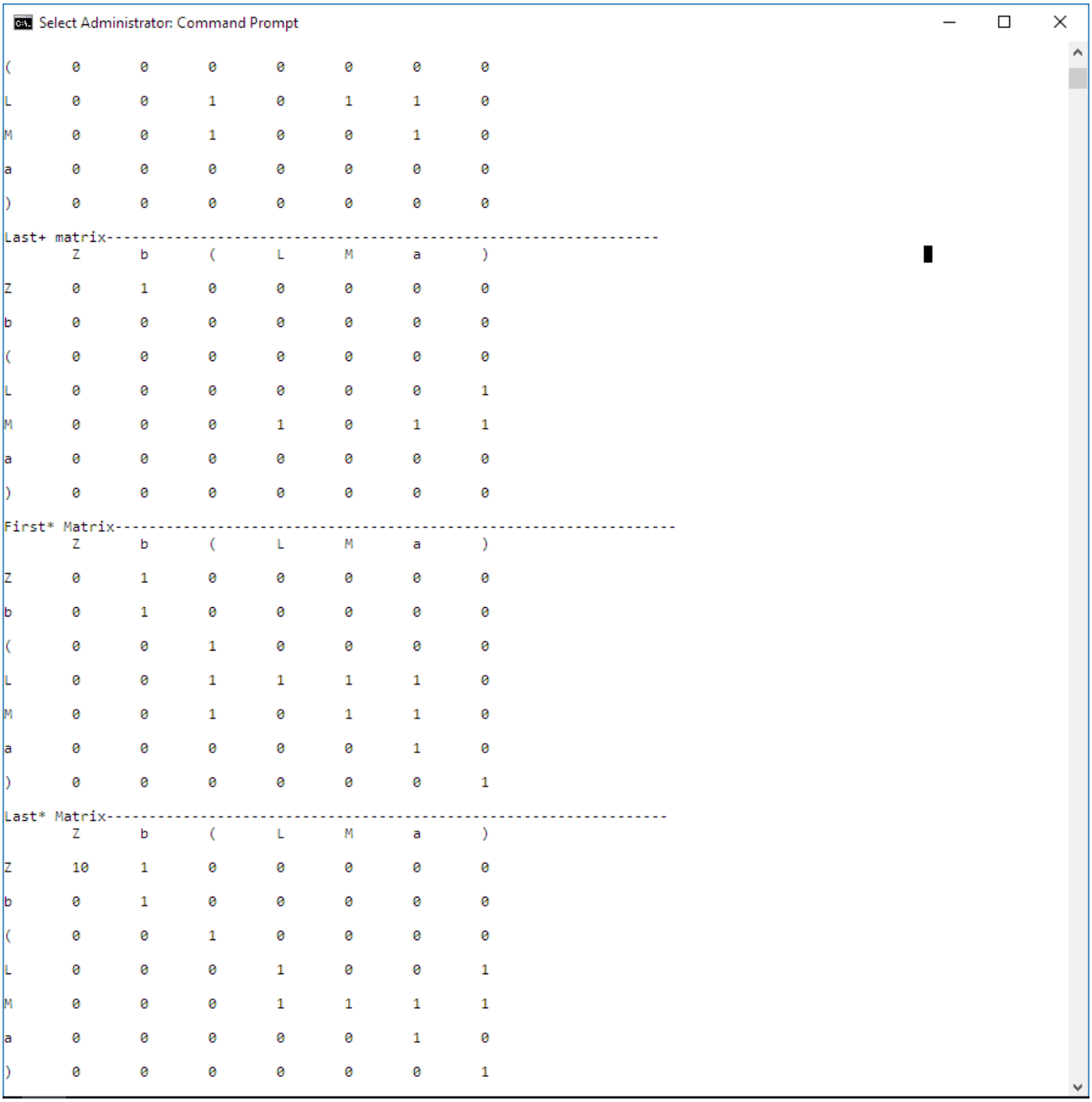
```
        if (prod[i].substring(l, l + 1).equals(firstm[j + 1][0]) &&
         prod[i].substring(l + 1, l + 2).equals(firstm[0][k + 1]))
         equalm[j + 1][k + 1] += "1";}}}}}catch (StringIndexOutOfBoundsException e) {}
  } } catch (NullPointerException e) {}
 System.out.println("Equal matrix----------------------------------------------------------------");
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (equalm[i + 1][j + 1] == "")
    equalm[i + 1][j + 1] += "0";
   System.out.print(equalm[i][j] + "\t");
  }
  System.out.println("\n");}}
 public void lessthan() {
  int temp, tempm;
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    less[i][j] = "";}}
  for (i = 0; i < ct; i++) {
   less[i + 1][0] += tempt.substring(i, i + 1);
  }
  for (j = 0; j < ct; j++) {
   less[0][j + 1] += tempt.substring(j, j + 1);
  }
  for (i = 1; i <= ct; i++) {
   for (j = 1; j <= ct; j++) {
    tempm = 0;
    for (k = 1; k <= ct; k++) {
     temp = Integer.parseInt(equalm[i][k]) & Integer.parseInt(firstp[k][j]);
     tempm = tempm | temp;
    }
    if (tempm == 1)
     less[i][j] += "1";}
  System.out.println("Lessthan matrix-------------------------------------------------------------");
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    if (less[i + 1][j + 1] == "")
     less[i + 1][j + 1] += "0";
    System.out.print(less[i][j] + "\t");
   }
   System.out.println("\n");}}
 public void greaterthan() {
  int temp, tempm;
  String temps[][] = new String[50][50];
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    greater[i][j] = "";}}
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    temps[i][j] = "";}}
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
```

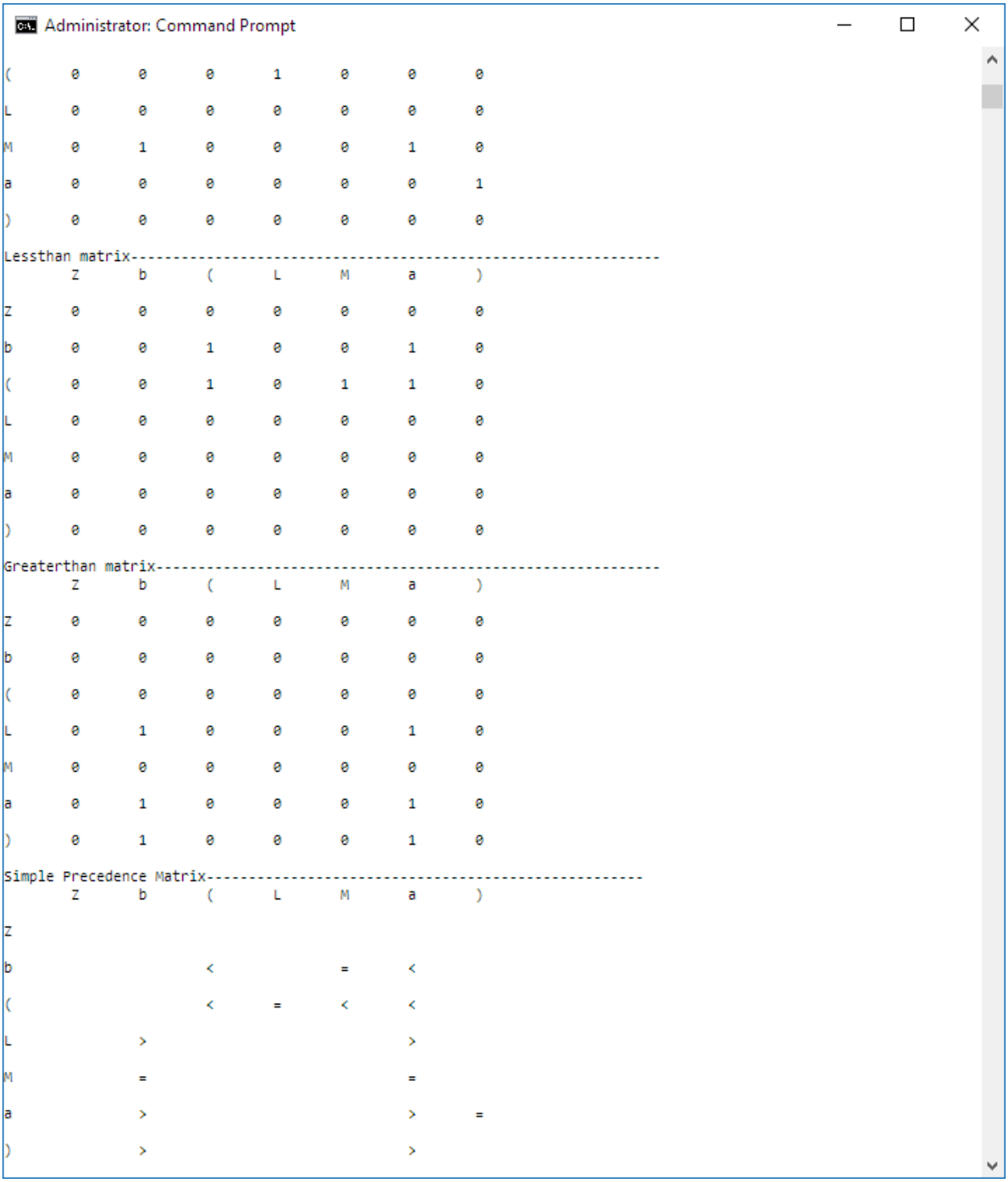
```
  lastt[i][j] = "";}}
 for (i = 0; i < ct; i++) {
  lastt[i + 1][0] += tempt.substring(i, i + 1);
 }
 for (j = 0; j < ct; j++) {
  lastt[0][j + 1] += tempt.substring(j, j + 1);
 }
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   lastt[i + 1][j + 1] += lastp[j + 1][i + 1];}}
 for (i = 0; i < ct; i++) {
  greater[i + 1][0] += tempt.substring(i, i + 1);
 }
 for (j = 0; j < ct; j++) {
  greater[0][j + 1] += tempt.substring(j, j + 1);
 }
 for (i = 0; i < ct; i++) {
  temps[i + 1][0] += tempt.substring(i, i + 1);
 }
 for (j = 0; j < ct; j++) {
  temps[0][j + 1] += tempt.substring(j, j + 1);
 }
 for (i = 1; i <= ct; i++) {
  for (j = 1; j <= ct; j++) {
   tempm = 0;
   for (k = 1; k <= ct; k++) {
    temp = Integer.parseInt(lastt[i][k]) & Integer.parseInt(equalm[k][j]);
    tempm = tempm | temp;
   }
   if (tempm == 1)
    temps[i][j] += "1";}}
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (temps[i + 1][j + 1] == "")
    temps[i + 1][j + 1] += "0";}}
 for (i = 1; i <= ct; i++) {
  for (j = 1; j <= ct; j++) {
   tempm = 0;
   for (k = 1; k <= ct; k++) {
    temp = Integer.parseInt(temps[i][k]) & Integer.parseInt(firsts[k][j]);
    tempm = tempm | temp;
   }
   if (tempm == 1)
    greater[i][j] += "1";}}
 System.out.println("Greaterthan matrix----------------------------------------------------------");
 for (i = 0; i <= ct; i++) {
  for (j = 0; j <= ct; j++) {
   if (greater[i + 1][j + 1] == "")
    greater[i + 1][j + 1] += "0";
   System.out.print(greater[i][j] + "\t");
  }
```

```
  System.out.println("\n");}}
 public void spmmatrix() {
  int flag = 0;
  for (i = 0; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    simplepm[i][j] = "";}}
  for (i = 0; i < ct; i++) {
   simplepm[i + 1][0] += tempt.substring(i, i + 1);
  }
  for (j = 0; j < ct; j++) {
   simplepm[0][j + 1] += tempt.substring(j, j + 1);
  }
  for (i = 1; i <= ct; i++) {
   for (j = 1; j <= ct; j++) {
    if (equalm[i][j].equals("1"))
     simplepm[i][j] += "=";
    if (less[i][j].equals("1"))
     simplepm[i][j] += "<";
    if (greater[i][j].equals("1"))
     simplepm[i][j] += ">";}}
  for (i = 1; i <= ct; i++) {
   for (j = 0; j <= ct; j++) {
    if (simplepm[i][j].length() > 1)
     flag = 1;}}
  if (flag == 1) {
   System.out.println("The given matrix is not Simple Precedence Matrix-------");
  } else {
   System.out.println("Simple Precedence Matrix----------------------------------------------------");
   for (i = 0; i <= ct; i++) {
    for (j = 0; j <= ct; j++) {
     System.out.print(simplepm[i][j] + "\t");
    }
    System.out.println("\n");}}}
 public static void main(String arr[]) {
  SPM s = new SPM();
  s.getproduction();
  s.first();
  s.last();
  s.firstplus();
  s.lastplus();
  s.firststar();
  s.laststar();
  s.equal();
  s.lessthan();
  s.greaterthan();
  s.spmmatrix();
 }
}
```

OUTPUT


Administrator: Command Prompt
D:\Stuff\CODE>java SPM
Enter The Production
Enter q to quit
Z-bMb
M-(L
M-a
L-Ma)
q
Produstions are:
Z-bMb
M-(L
M-a
L-Ma)
Zb(LMa)
First matrix----------
Z b (L M a)
Z 0 1 0 0 0 0 0
b 0 0 0 0 0 0 0
(0 0 0 0 0 0 0
L 0 0 0 0 1 0 0
M 0 0 1 0 0 1 0
a 0 0 0 0 0 0 0
) 0 0 0 0 0 0 0
Last matrix----------
Z b (L M a)
Z 0 1 0 0 0 0 0
b 0 0 0 0 0 0 0
(0 0 0 0 0 0 0
L 0 0 0 0 0 0 1
M 0 0 0 1 0 1 0
a 0 0 0 0 0 0 0
) 0 0 0 0 0 0 0
First+ matrix----------
Z b (L M a)
Z 0 1 0 0 0 0 0
b 0 0 0 0 0 0 0
(0 0 0 0 0 0 0
L 0 0 1 0 1 1 0
M 0 0 1 0 0 1 0


Select Administrator: Command Prompt
(0 0 0 0 0 0 0
L 0 0 1 0 1 1 0
M 0 0 1 0 0 1 0
a 0 0 0 0 0 0 0
) 0 0 0 0 0 0 0
Last+ matrix----------
Z b (L M a)
Z 0 1 0 0 0 0 0
b 0 0 0 0 0 0 0
(0 0 0 0 0 0 0
L 0 0 0 0 0 0 1
M 0 0 0 1 0 1 1
a 0 0 0 0 0 0 0
) 0 0 0 0 0 0 0
First* Matrix----------
Z b (L M a)
Z 0 1 0 0 0 0 0
b 0 1 0 0 0 0 0
(0 0 1 0 0 0 0
L 0 0 1 1 1 1 0
M 0 0 1 0 1 1 0
a 0 0 0 0 0 1 0
) 0 0 0 0 0 0 1
Last* Matrix----------
Z b (L M a)
Z 10 1 0 0 0 0 0
b 0 1 0 0 0 0 0
(0 0 1 0 0 0 0
L 0 0 0 1 0 0 1
M 0 0 0 1 1 1 1
a 0 0 0 0 0 1 0
) 0 0 0 0 0 0 1


Administrator: Command Prompt
(0 0 0 1 0 0 0
L 0 0 0 0 0 0 0
M 0 1 0 0 0 1 0
a 0 0 0 0 0 0 1
) 0 0 0 0 0 0 0
Lessthan matrix----------
Z b (L M a)
Z 0 0 0 0 0 0 0
b 0 0 1 0 0 1 0
(0 0 1 0 1 1 0
L 0 0 0 0 0 0 0
M 0 0 0 0 0 0 0
a 0 0 0 0 0 0 0
) 0 0 0 0 0 0 0
Greaterthan matrix----------
Z b (L M a)
Z 0 0 0 0 0 0 0
b 0 0 0 0 0 0 0
(0 0 0 0 0 0 0
L 0 1 0 0 0 1 0
M 0 0 0 0 0 0 0
a 0 1 0 0 0 1 0
) 0 1 0 0 0 1 0
Simple Precedence Matrix----------
Z b (L M a)
Z
b < = <
(< = < <
L > >
M = =
a > > =
) > >

Practical 4

Write a program to illustrate the generation on OPM for the input operator Grammar.

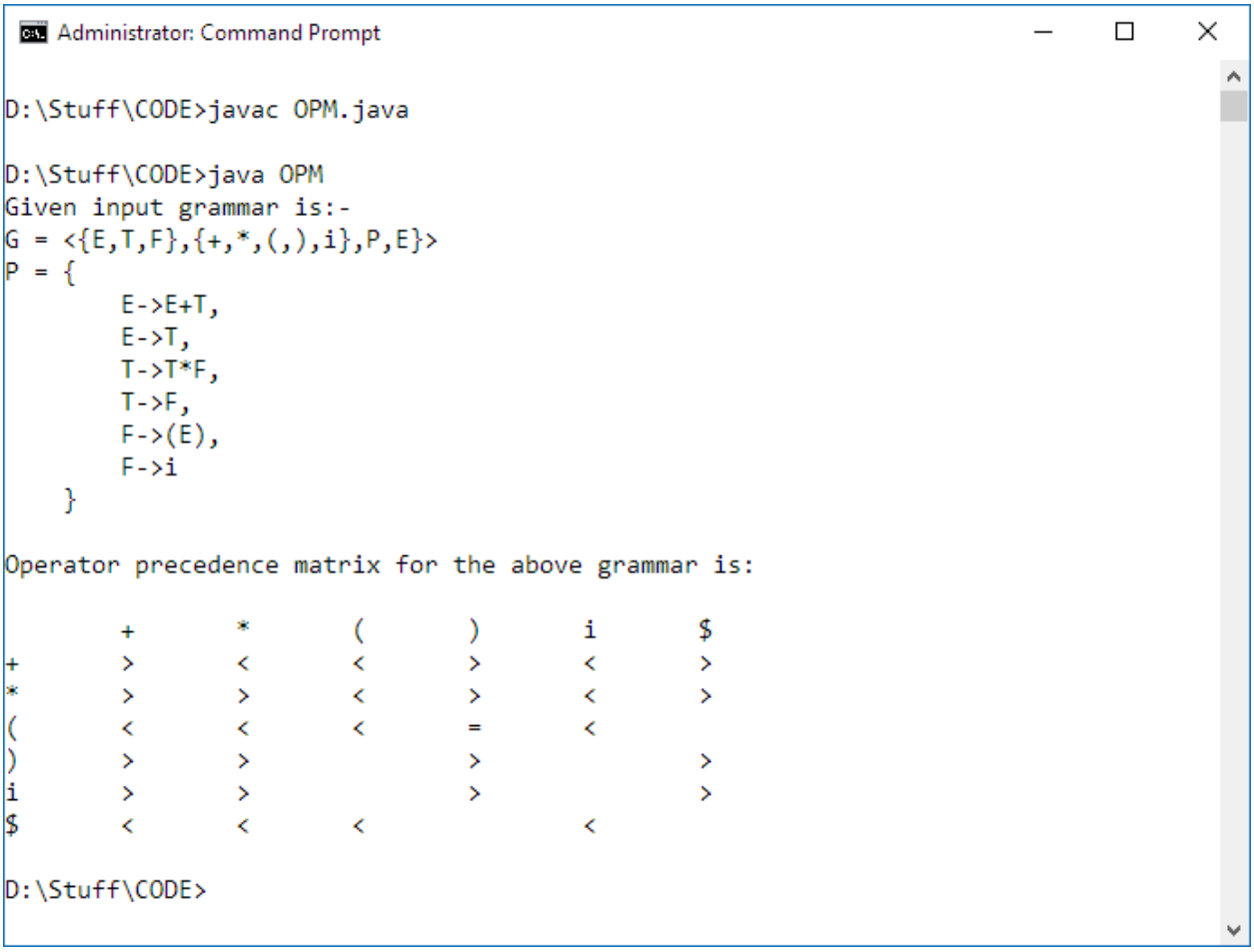
Code:

```
public class OPM {
 public static int i, j, k, ind, ind1;
 public static String[] prod = {
  "E->E+T",
  "E->T",
  "T->T*F",
  "T->F",
  "F->(E)",
  "F->i"
 };
 public static String syms = "ETF+*()i", nt = "ETF", t = "+*()i";
 public static final int LEN = syms.length(), NLEN = nt.length(), TLEN = t.length();
 public static int[][] f = new int[LEN][LEN];
 public static int[][] l = new int[LEN][LEN];
 public static char[][] opm = new char[TLEN + 1][TLEN + 1];
 public static void main(String[] args) {
   System.out.println("Given input grammar is:-");
   printGrammar(); // print grammer method call
   for (String p: prod) {
    f[syms.indexOf(p.charAt(0))][syms.indexOf(p.charAt(3))] = 1; // charAt =This method returns the character
located at the String's specified index. The string indexes start from zero.
    l[syms.indexOf(p.charAt(0))][syms.indexOf(p.charAt(p.length() - 1))] = 1;
    if (p.length() > 4 && t.contains("" + p.charAt(4))) {
     f[syms.indexOf(p.charAt(0))][syms.indexOf(p.charAt(4))] = 1;
     l[syms.indexOf(p.charAt(0))][syms.indexOf(p.charAt(4))] = 1;}}
   f = getWarshallClosure(f); // warshall closure method call
   l = getWarshallClosure(l); // warshall closure method call
   System.out.println("\nOperator precedence matrix for the above grammar is: \n");
   t = t + "$";
   for (i = 0; i < TLEN; i++) {
    if (f[0][NLEN + i] != 0)
     opm[TLEN][i] = '<';
    if (l[0][NLEN + i] != 0)
     opm[i][TLEN] = '>';
   }
   for (String p: prod) {
    String rhs = p.substring(3, p.length()), x, b, c = "";
    if (rhs.length() >= 2)
     c = "" + rhs.charAt(2);
    if (rhs.length() > 1) {
     x = "" + rhs.charAt(0);
     b = "" + rhs.charAt(1);
     if (t.contains(x) && t.contains(b))
      opm[t.indexOf(x)][t.indexOf(b)] = '=';
     if (t.contains(x) && nt.contains(b))
      if (t.contains(c))
       opm[t.indexOf(x)][t.indexOf(c)] = '=';
```

```
    if (nt.contains(x) && t.contains(b)) {
     ind = nt.indexOf(x);
     ind1 = t.indexOf(b);
     for (i = 0; i < TLEN; i++)
      if (l[ind][NLEN + i] != 0)
       opm[i][ind1] = '>';
    } else if (nt.contains(b) && t.contains(c)) {
     ind = nt.indexOf(b);
     ind1 = t.indexOf(c);
     for (i = 0; i < TLEN; i++)
      if (l[ind][NLEN + i] != 0)
       opm[i][ind1] = '>';
    }
    if (t.contains(x) && nt.contains(b)) {
     ind = t.indexOf(x);
     ind1 = nt.indexOf(b);
     for (i = 0; i < TLEN; i++)
      if (f[ind1][NLEN + i] != 0)
       opm[ind][i] = '<';
    } else if (t.contains(b) && nt.contains(c)) {
     ind = t.indexOf(b);
     ind1 = nt.indexOf(c);
     for (i = 0; i < TLEN; i++)
      if (f[ind1][NLEN + i] != 0)
       opm[ind][i] = '<';}}}
  for (i = 0; i <= TLEN; i++)
   System.out.print("\t" + t.charAt(i));
  System.out.println();
  for (i = 0; i <= TLEN; i++) {
   System.out.print(t.charAt(i) + "\t");
   for (j = 0; j <= TLEN; j++)
    System.out.print(opm[i][j] + "\t");
   System.out.println();}}
 // warshall closure method
 public static int[][] getWarshallClosure(int[][] a) {
  for (i = 0; i < a.length; i++) {
   for (j = 0; j < a.length; j++) {
    if (a[j][i] == 1) {
     for (k = 0; k < a.length; k++) {
      a[j][k] = a[j][k] | a[i][k];}}}}
  return a;
 }
 //print grammer method
 public static void printGrammar() {
  String grammar = "G = <{" + nt.charAt(0) + ",";
  for (i = 1; i < nt.length() - 1; i++)
   grammar += nt.charAt(i) + ",";
  grammar += nt.charAt(nt.length() - 1) + "},{" + t.charAt(0) + ",";
  for (i = 1; i < t.length() - 1; i++)
   grammar += t.charAt(i) + ",";
  grammar += t.charAt(t.length() - 1) + "},P," + nt.charAt(0) + "}>\nP = {\n\t" + prod[0] + ",";
```

```
 for (i = 1; i < prod.length - 1; i++)
  grammar += "\n\t" + prod[i] + ",";
 System.out.println(grammar + "\n\t" + prod[prod.length - 1] + "\n    }");
 }
}
```

OUTPUT



Practical 5
Write a code to generate the DAG for the input arithmetic expression.

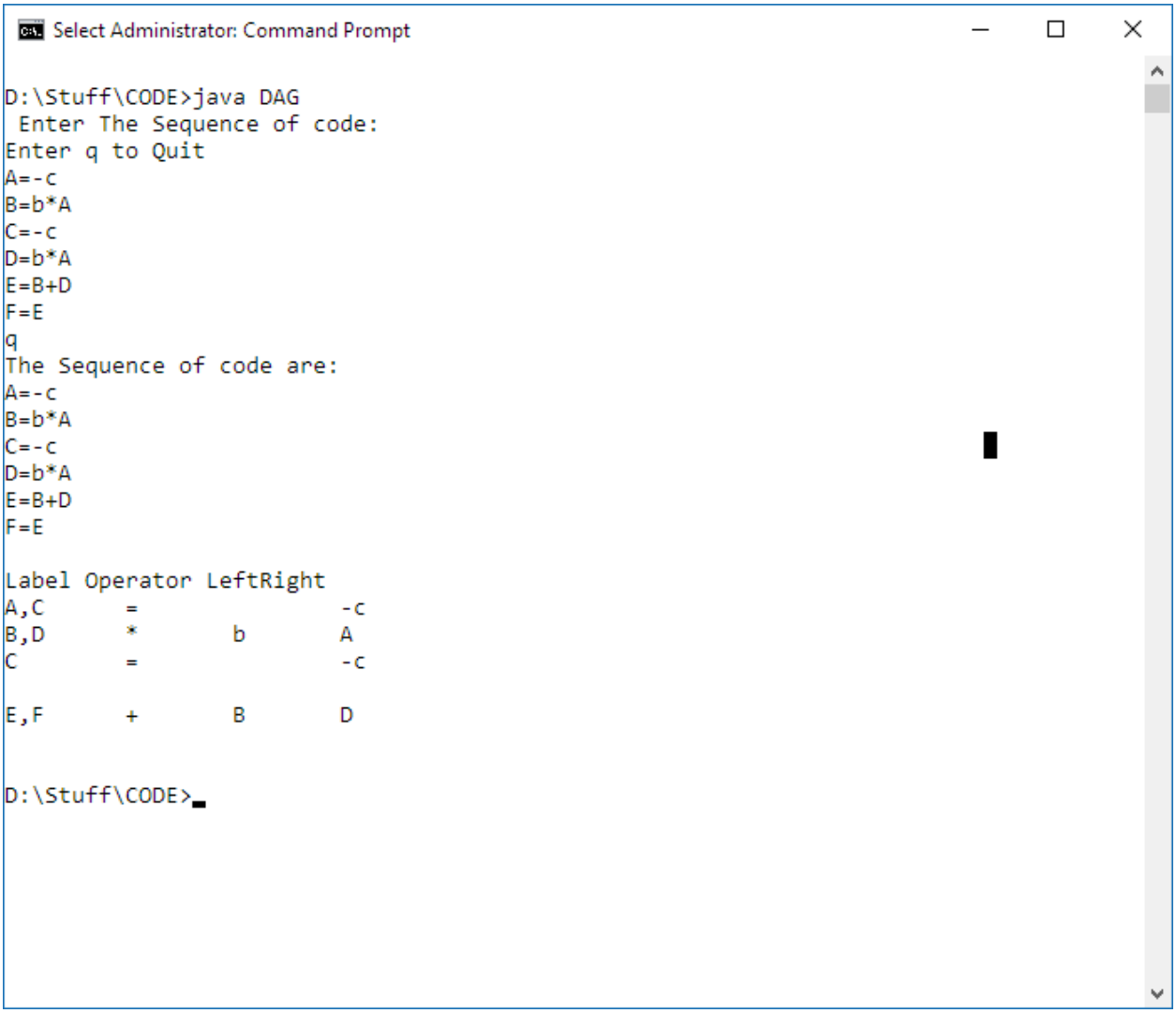
Code:

```
import java.io.*;
public class DAG {
 int i, j, k;
 int count = -1, flag = 0;
 String str[] = new String[10];
 String table[][] = new String[10][10];
 public DAG() {
  try {
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println(" Enter The Sequence of code:");
   System.out.println("Enter q to Quit");
   for (i = 0; i < str.length; i++) {
    str[i] = br.readLine();
    if (str[i].equals("q"))
     break;
    else
     count++;
   }
   System.out.println("The Sequence of code are:");
   for (i = 0; i < str.length; i++) {
    if (str[i].equals("q"))
     break;
    System.out.println(str[i]);}}catch (IOException e) {}
  //System.out.println("="+count);
 }
 public void tablestruct() {
  for (i = 0; i < str.length; i++) {
   for (j = 0; j < str.length; j++) {
    table[i][j] = "";}}
  try

  {
   for (i = 0; i <= count; i++) {
    for (j = 0; j <= count; j++) {
     if (str[i].length() == 4) {
      if (j == 0)
       table[i][j] = str[i].substring(0, 1);
      if (j == 1)
       table[i][j] = str[i].substring(1, 2);
      if (j == 3)
       table[i][j] = str[i].substring(2);
     }
     if (str[i].length() == 3) {
      if (j == 0)
       table[i][j] = str[i].substring(0, 1);
      if (j == 1)
```

```
      table[i][j] = str[i].substring(1, 2);
     if (j == 3)
      table[i][j] = str[i].substring(2);
    }
    if (str[i].length() == 5) {
     if (j == 0)
      table[i][j] = str[i].substring(0, 1);
     if (j == 1)
      table[i][j] = str[i].substring(3, 4);
     if (j == 2)
      table[i][j] = str[i].substring(2, 3);
     if (j == 3)
      table[i][j] = str[i].substring(4);}}}}catch (NullPointerException e) {}
  for (i = 0; i <= count; i++) {
   for (j = i + 1; j <= count; j++) {
    if (str[i].length() == 4 && str[j].length() == 4) {
     if (str[i].substring(2, 4).equals(str[j].substring(2, 4))) {
      table[i][0] = table[i][0].concat("," + str[j].substring(0, 1));
      //for(k=0;k<=count;k++)
      // table[j][k]="";}}
    if (str[i].length() == 5 && str[j].length() == 5) {
     if (str[i].substring(2, 5).equals(str[j].substring(2, 5))) {
      table[i][0] = table[i][0].concat("," + str[j].substring(0, 1));
      for (k = 0; k <= count; k++)
       table[j][k] = "";}}}
   if (str[i].length() == 3) {
    if (i == count) {
     for (j = count - 1; j >= 0; j--) {
      if (str[i].substring(2, 3).equals(str[j].substring(0, 1))) {
       table[j][0] = table[j][0].concat("," + str[i].substring(0, 1));
       for (k = 0; k <= count; k++)
        table[i][k] = "";}}}else {
     for (j = i + 1; j <= count; j++) {
      if (str[i].substring(2, 3).equals(str[j].substring(0, 1))) {
       table[i][0] = table[i][0].concat("," + str[j].substring(0, 1));
       for (k = 0; k <= count; k++)
        table[j][k] = "";}}}}}
  System.out.println();
  System.out.print("Label" + " " + "Operator" + " " + "Left" + "" + "Right");
  System.out.println();
  for (i = 0; i <= count; i++) {
   for (j = 0; j <= count; j++) {
    System.out.print(table[i][j] + "\t ");}
   System.out.println();}}
 public static void main(String arg[]) {
  DAG d = new DAG();
  d.tablestruct();
 }
}
```

OUTPUT


Select Administrator: Command Prompt
D:\Stuff\CODE>java DAG
Enter The Sequence of code:
Enter q to Quit
A=-c
B=b*A
C=-c
D=b*A
E=B+D
F=E
q
The Sequence of code are:
A=-c
B=b*A
C=-c
D=b*A
E=B+D
F=E
Label Operator LeftRight
A,C = -c
B,D * b A
C = -c
E,F + B D
D:\Stuff\CODE>

Practical 6

Write a program to demonstrate loop unrolling and loop splitting for the given code sequence containing loop.

Code:

```
public class loop_unrolling {
 public static void main(String[] args) {
  int[] array1 = new int[5];
  long t1 = System.currentTimeMillis();
  // Version 1: assign elements in a loop.
  for (int i = 0; i < 10000000; i++) {
   for (int x = 0; x < array1.length; x++) {
    array1[x] = x;
   }
  }
  long t2 = System.currentTimeMillis();
  // Version 2: unroll the loop and use a list of statements.
  for (int i = 0; i < 10000000; i++) {
   array1[0] = 0;
   array1[1] = 1;
   array1[2] = 2;
   array1[3] = 3;
   array1[4] = 4;
  }
  long t3 = System.currentTimeMillis();
  // ... Times.
  System.out.println("Time taken by processor before loop unrolling:--> " + (t2 - t1));
  System.out.println("Time taken by processor after loop unrolling:--> " + (t3 - t2));
 }
}
```

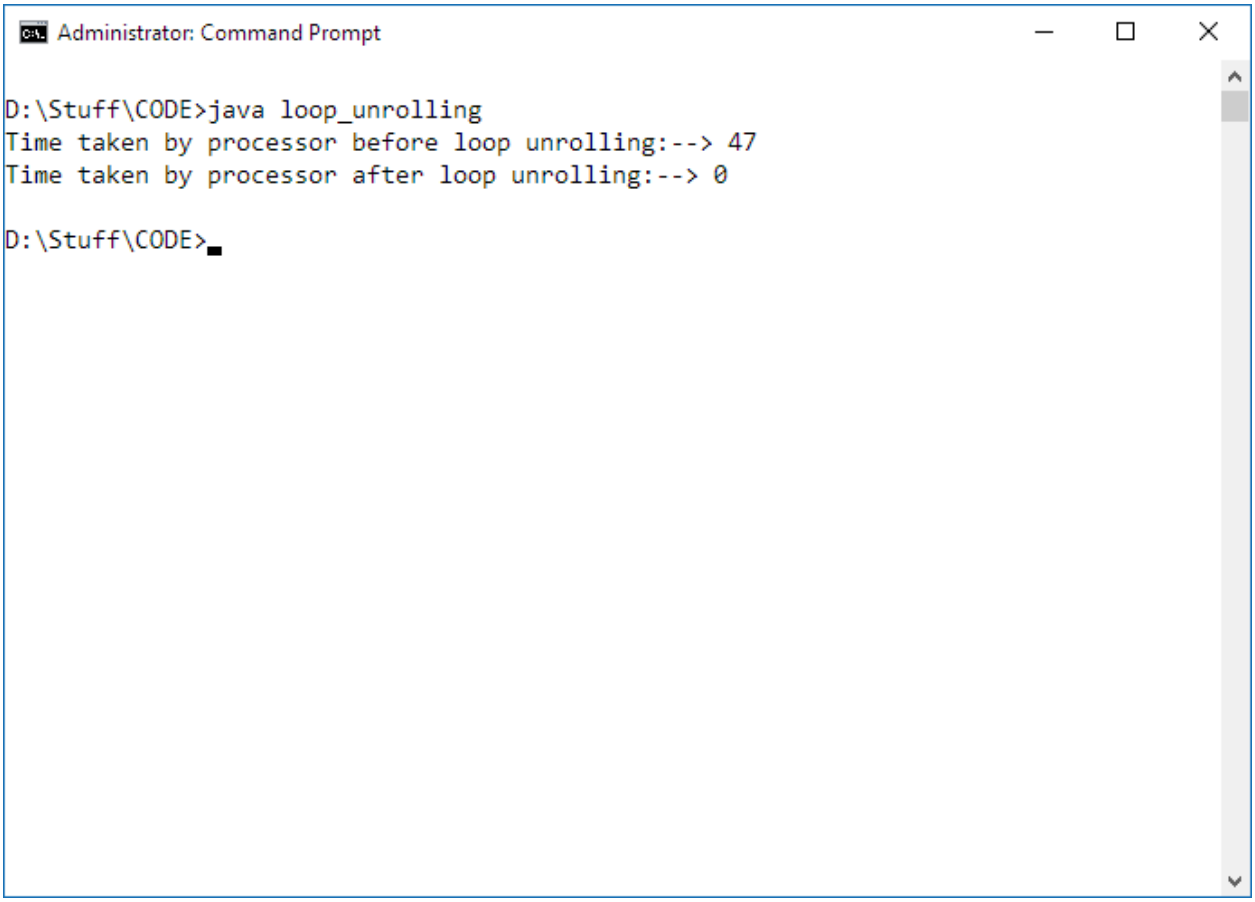
Loop_jamming.java

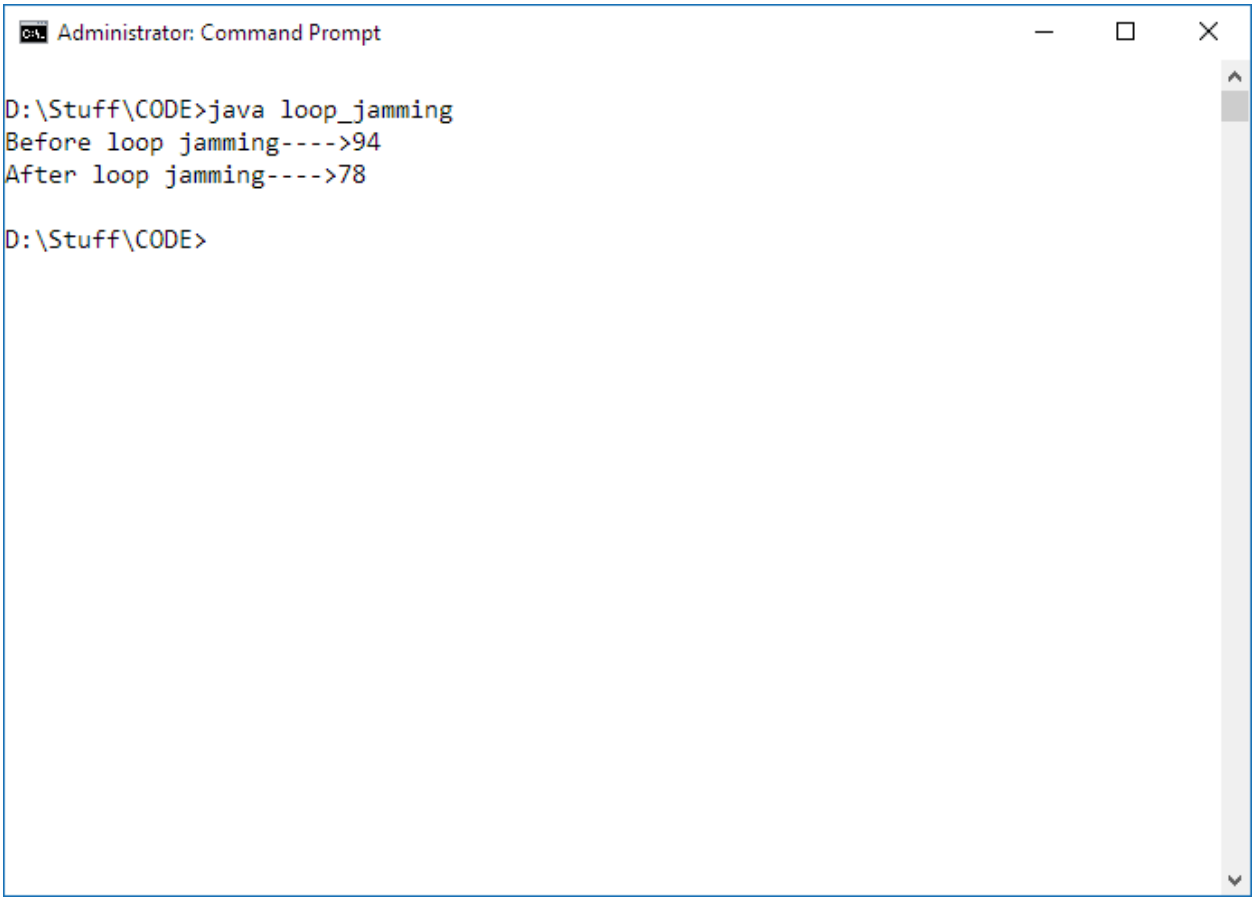
```
public class loop_jamming {
 public static void main(String[] args) {
  int[] array1 = {  10,  20,  30  };
  int[] array2 = {  20,  10,  30  };
  int[] array3 = {  40,  40,  10  };
  long t1 = System.currentTimeMillis();
  // Version 1: loop over each array separately.
  for (int i = 0; i < 10000000; i++) {
   int sum = 0;
   for (int x = 0; x < array1.length; x++) {
    sum += array1[x];
   }
   for (int x = 0; x < array2.length; x++) {
    sum += array2[x];
   }
   for (int x = 0; x < array3.length; x++) {
```

```
   sum += array3[x];
  }
  if (sum != 210) {
   System.out.println(false);
  }

 }
 long t2 = System.currentTimeMillis();
 // Version 2: jam loops together.
 for (int i = 0; i < 10000000; i++) {
  int sum = 0;
  for (int x = 0; x < array1.length; x++) {
   sum += array1[x];
   sum += array2[x];
   sum += array3[x];
  }
  if (sum != 210) {
   System.out.println(false);
  }
 }
 long t3 = System.currentTimeMillis();
 // ... Times.
 System.out.println("Before loop jamming----->" + (t2 - t1));
 System.out.println("After loop jamming----->" + (t3 - t2));
}
}
```

OUTPUT


Administrator: Command Prompt
D:\Stuff\CODE>java loop_unrolling
Time taken by processor before loop unrolling:--> 47
Time taken by processor after loop unrolling:--> 0
D:\Stuff\CODE>


Administrator: Command Prompt
D:\Stuff\CODE>java loop_jamming
Before loop jamming---->94
After loop jamming---->78
D:\Stuff\CODE>