Practical No 3:

Aim: Write a program to illustrate the generation on SPM for the input grammar.

Program Code:

SPM_1.java

```
public class SPM_1 {

   SPM_1() {
   }

   void PrintProduction(String Prod[]) {

      for (int i = 0; i < Prod.length; i++) {
         System.out.println("\t" + Prod[i]);
      }
      //System.out.println("\n");
   }

   void displayMatrix(int Matrix[][], String strString) {
      System.out.print("\t");
      for (int i = 0; i < strString.length(); i++) {
         System.out.print("   " + strString.charAt(i));
      }
      for (int i = 0; i < 7; i++) {
         System.out.println();
         System.out.print("       " + strString.charAt(i) + "  ");
         for (int j = 0; j < 7; j++) {
            System.out.print(Matrix[i][j] + "   ");
         }
      }
   }

   void displayMatrix1(String matrix[][], String strString) {
      System.out.print("\t");
      for (int i = 0; i < strString.length(); i++) {
         System.out.print("   " + strString.charAt(i));
      }
      for (int i = 0; i < 7; i++) {
         System.out.println();
         System.out.print("       " + strString.charAt(i) + "  ");
         for (int j = 0; j < 7; j++) {
            System.out.print(matrix[i][j] + "   ");
         }
      }
   }

}
```

SPM_2.java

```
public class SPM_2 {

   static int i, j, k, l = 0;
   static SPM_1 s = new SPM_1();
   static String Prod[] = new String[4];
   static String NT[] = {"Z", "M", "L"};
   static String T[] = {"a", "b", "(", ")"};
   static String first[] = new String[4];
   static String last[] = new String[4];
   static String equal[] = new String[5];

   static int first_matrix[][] = new int[NT.length + T.length][NT.length + T.length];
   static int last_matrix[][] = new int[NT.length + T.length][NT.length + T.length];
   static int equal_matrix[][] = new int[NT.length + T.length][NT.length + T.length];
   static int SPM[][] = new int[NT.length + T.length][NT.length + T.length];

   static int A[][] = new int[NT.length + T.length][NT.length + T.length];
   static int B[][] = new int[NT.length + T.length][NT.length + T.length];
   static int C[][] = new int[NT.length + T.length][NT.length + T.length];
   static int D[][] = new int[NT.length + T.length][NT.length + T.length];
   static int E[][] = new int[NT.length + T.length][NT.length + T.length];
   static int I[][] = new int[NT.length + T.length][NT.length + T.length];

   static int TlastPlus_matrix[][] = new int[NT.length + T.length][NT.length + T.length];
   static int lessthan_matrix[][] = new int[NT.length + T.length][NT.length + T.length];
   static int greaterthan_matrix[][] = new int[NT.length + T.length][NT.length + T.length];
   static String spm[][] = new String[NT.length + T.length][NT.length + T.length];
   static String strString = "ZbMLa()";

   public static void SPM() {
      for (i = 0; i < NT.length + T.length; i++) {
         for (j = 0; j < NT.length + T.length; j++) {
            if (lessthan_matrix[i][j] == 1) {
               spm[i][j] = "<";
            } else if (greaterthan_matrix[i][j] == 1) {
               spm[i][j] = ">";
            } else if (equal_matrix[i][j] == 1) {
               spm[i][j] = "=";
            } else {
               spm[i][j] = "0";
            }
         }
      }
   }

   public static void FirstPlus(int matrix[][]) {
      System.out.println("\n\n" + "first+ matrix is:");
      System.out.println();
      for (i = 0; i < NT.length + T.length; i++) {
         for (j = 0; j < NT.length + T.length; j++) {
            A[i][j] = matrix[i][j];
         }
```

```
    }
    for (i = 1; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        if (A[j][i] == 1) {
          for (k = 1; k < NT.length + T.length; k++) {
            A[j][k] = A[j][k] | A[i][k];
          }
        }
      }
    }
    s.displayMatrix(A, strString);
}

public static void FirstStar(int first_matrix[][]) {
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        if (i == j) {
          I[i][j] = 1;
        } else {
          I[i][j] = 0;
        }
      }
    }
    System.out.println("\n\n" + "first*_matrix is;");
    System.out.println();
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        C[i][j] = A[i][j] | I[i][j];
      }
    }
    s.displayMatrix(C, strString);
}

public static void LastPlus(int matrix[][]) {
    System.out.println("\n" + "last+ matrix is:");
    System.out.println();
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        B[i][j] = last_matrix[i][j];
      }
    }
    for (i = 1; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        if (B[j][i] == 1) {
          for (k = 1; k < NT.length + T.length; k++) {
            B[j][k] = B[j][k] | B[i][k];
          }
        }
      }
    }
    s.displayMatrix(B, strString);
}

public static void LastStar(int matrix[][]) {
```

```
    System.out.println("\n" + "last*_matrix is;");
    System.out.println();
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        E[i][j] = B[i][j] | I[i][j];
      }
    }
    s.displayMatrix(E, strString);
}

public static void EqualMatrix() {
    //Finding the elements of equal
    int pos = Prod[0].indexOf(">");
    for (i = 0; i < Prod.length; i++) {
      String str = Prod[i].substring(pos + 1);
      if (str.length() >= 2) {
        for (j = 0; j < str.length() - 1; j++) {
          equal[l] = str.charAt(j) + "" + str.charAt(j + 1);
          l++;
        }
      }
    }
    //Displaying of equal elements
    System.out.println();
    System.out.println("\nequal Elements ::");
    s.PrintProduction(equal);
    for (i = 0; i < equal_matrix.length; i++) {
      for (j = 0; j < equal_matrix.length; j++) {
        equal_matrix[i][j] = 0;
      }
    }
    for (i = 0; i < strString.length(); i++) {
      for (j = 0; j < strString.length(); j++) {
        for (k = 0; k < equal.length; k++) {
          if (strString.charAt(i) == equal[k].charAt(0) && strString.charAt(j) == equal[k].charAt(1)) {
            equal_matrix[i][j] = 1;
          }
        }
      }
    }
    s.displayMatrix(equal_matrix, strString);
}

public static void LessThanMatrix() {
    System.out.println();
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        lessthan_matrix[i][j] = equal_matrix[i][0] * A[0][j]
            + equal_matrix[i][1] * A[1][j]
            + equal_matrix[i][2] * A[2][j]
            + equal_matrix[i][3] * A[3][j]
            + equal_matrix[i][4] * A[4][j]
            + equal_matrix[i][5] * A[5][j]
            + equal_matrix[i][6] * A[6][j];
```

```
      }
    }
}

public static void TransposeLastPlus() {
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        TlastPlus_matrix[i][j] = B[j][i];
      }
    }
}

public static void GreaterThanMatrix() {
    System.out.println("\n" + "greater_than matrix is;");
    System.out.println();
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        D[i][j] = TlastPlus_matrix[i][0] * equal_matrix[0][j]
              + TlastPlus_matrix[i][1] * equal_matrix[1][j]
              + TlastPlus_matrix[i][2] * equal_matrix[2][j]
              + TlastPlus_matrix[i][3] * equal_matrix[3][j]
              + TlastPlus_matrix[i][4] * equal_matrix[4][j]
              + TlastPlus_matrix[i][5] * equal_matrix[5][j]
              + TlastPlus_matrix[i][6] * equal_matrix[6][j];
      }
    }
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        greaterthan_matrix[i][j] = D[i][0] * C[0][j]
              + D[i][1] * C[1][j]
              + D[i][2] * C[2][j]
              + D[i][3] * C[3][j]
              + D[i][4] * C[4][j]
              + D[i][5] * C[5][j]
              + D[i][6] * C[6][j];
      }
    }
    s.displayMatrix(greaterthan_matrix, strString);
}

public static void spm_Matrix() {
    System.out.println("\n" + "SPM matrix is;");
    System.out.println();
    for (i = 0; i < NT.length + T.length; i++) {
      for (j = 0; j < NT.length + T.length; j++) {
        SPM[i][j] = lessthan_matrix[i][j] | greaterthan_matrix[i][j] | equal_matrix[i][j];
      }
    }
    s.displayMatrix(SPM, strString);
}

public static void main(String[] args) {

    Prod[0] = "Z->bMb";
```

```
Prod[1] = "M->(L";
Prod[2] = "M->a";
Prod[3] = "L->Ma)";
//Displaying of productins.
System.out.println(" Productions are::");
s.PrintProduction(Prod);
//Finding Elements of First
for (i = 0; i < Prod.length; i++) {
   first[i] = Prod[i].charAt(0) + "" + Prod[i].charAt(3);
}
//Displaying of First elements
System.out.println("First Elements ::");
s.PrintProduction(first);
System.out.println("First Matrix ::");
for (i = 0; i < first_matrix.length; i++) {
   for (j = 0; j < first_matrix.length; j++) {
      first_matrix[i][j] = 0;
   }
}
for (i = 0; i < strString.length(); i++) {
   for (j = 0; j < strString.length(); j++) {
      for (k = 0; k < first.length; k++) {
         if (strString.charAt(i) == first[k].charAt(0) && strString.charAt(j) == first[k].charAt(1)) {
            first_matrix[i][j] = 1;
         }
      }
   }
}
s.displayMatrix(first_matrix, strString);
FirstPlus(first_matrix);
FirstStar(first_matrix);
//Finding the elements of Last
for (i = 0; i < Prod.length; i++) {
   last[i] = Prod[i].charAt(0) + "" + Prod[i].charAt(Prod[i].length() - 1);
}
//Displaying of Last elements
System.out.println();
System.out.println("\n Last Elements ::");
s.PrintProduction(last);
System.out.println("Last Matrix ::");
for (i = 0; i < last_matrix.length; i++) {
   for (j = 0; j < last_matrix.length; j++) {
      last_matrix[i][j] = 0;
   }
}
for (i = 0; i < strString.length(); i++) {
   for (j = 0; j < strString.length(); j++) {
      for (k = 0; k < last.length; k++) {
         if (strString.charAt(i) == last[k].charAt(0) && strString.charAt(j) == last[k].charAt(1)) {
            last_matrix[i][j] = 1;
         }
      }
   }
}
```

```
        s.displayMatrix(last_matrix, strString);
        LastPlus(last_matrix);
        LastStar(last_matrix);
        //Displaying of equal elements
        //Displaying less than matrix
        EqualMatrix();
        System.out.println();
        System.out.println("\n" + "less_than matrix is;");
        LessThanMatrix();
        s.displayMatrix(lessthan_matrix, strString);
        TransposeLastPlus();
        GreaterThanMatrix();
        spm_Matrix();
        System.out.println("\n The SPM Matrix is:\n ");
        SPM();
        s.displayMatrix1(spm, strString);
    }

}
```

Output:

```
Productions are::
        Z->bMb
        M->(L
        M->a
        L->Ma)
```

```
First Elements ::
         Zb
         M(
         Ma
         LM
First Matrix ::
            Z   b   M   L   a   (   )
         Z  0   1   0   0   0   0   0
         b  0   0   0   0   0   0   0
         M  0   0   0   0   1   1   0
         L  0   0   1   0   0   0   0
         a  0   0   0   0   0   0   0
         (  0   0   0   0   0   0   0
         )  0   0   0   0   0   0   0

first+ matrix is:

            Z   b   M   L   a   (   )
         Z  0   1   0   0   0   0   0
         b  0   0   0   0   0   0   0
         M  0   0   0   0   1   1   0
         L  0   0   1   0   1   1   0
         a  0   0   0   0   0   0   0
         (  0   0   0   0   0   0   0
         )  0   0   0   0   0   0   0

first*_matrix is;

            Z   b   M   L   a   (   )
         Z  1   1   0   0   0   0   0
         b  0   1   0   0   0   0   0
         M  0   0   1   0   1   1   0
         L  0   0   1   1   1   1   0
         a  0   0   0   0   1   0   0
         (  0   0   0   0   0   1   0
         )  0   0   0   0   0   0   1
```

```
 Last Elements ::
        Zb
        ML
        Ma
        L)
Last Matrix ::
           Z   b   M   L   a   (   )
        Z  0   1   0   0   0   0   0
        b  0   0   0   0   0   0   0
        M  0   0   0   1   1   0   0
        L  0   0   0   0   0   0   1
        a  0   0   0   0   0   0   0
        (  0   0   0   0   0   0   0
        )  0   0   0   0   0   0   0
last+ matrix is:

           Z   b   M   L   a   (   )
        Z  0   1   0   0   0   0   0
        b  0   0   0   0   0   0   0
        M  0   0   0   1   1   0   1
        L  0   0   0   0   0   0   1
        a  0   0   0   0   0   0   0
        (  0   0   0   0   0   0   0
        )  0   0   0   0   0   0   0
last*_matrix is;

           Z   b   M   L   a   (   )
        Z  1   1   0   0   0   0   0
        b  0   1   0   0   0   0   0
        M  0   0   1   1   1   0   1
        L  0   0   0   1   0   0   1
        a  0   0   0   0   1   0   0
        (  0   0   0   0   0   1   0
        )  0   0   0   0   0   0   1
```

```
equal Elements ::
        bM
        Mb
        (L
        Ma
        a)
```

	Z	b	M	L	a	(	)
Z	0	0	0	0	0	0	0
b	0	0	1	0	0	0	0
M	0	1	0	0	1	0	0
L	0	0	0	0	0	0	0
a	0	0	0	0	0	0	1
(	0	0	0	1	0	0	0
)	0	0	0	0	0	0	0

less_than matrix is;

	Z	b	M	L	a	(	)
Z	0	0	0	0	0	0	0
b	0	0	0	0	1	1	0
M	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0
(	0	0	1	0	1	1	0
)	0	0	0	0	0	0	0

greater_than matrix is;

	Z	b	M	L	a	(	)
Z	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0
L	0	1	0	0	1	0	0
a	0	1	0	0	1	0	0
(	0	0	0	0	0	0	0
)	0	1	0	0	1	0	0

SPM matrix is;

	Z	b	M	L	a	(	)
Z	0	0	0	0	0	0	0
b	0	0	1	0	1	1	0
M	0	1	0	0	1	0	0
L	0	1	0	0	1	0	0
a	0	1	0	0	1	0	1
(	0	0	1	1	1	1	0
)	0	1	0	0	1	0	0

The SPM Matrix is:

	Z	b	M	L	a	(	)
Z	0	0	0	0	0	0	0
b	0	0	=	0	<	<	0
M	0	=	0	0	=	0	0
L	0	>	0	0	>	0	0
a	0	>	0	0	>	0	=
(	0	0	<	=	<	<	0
)	0	>	0	0	>	0	0