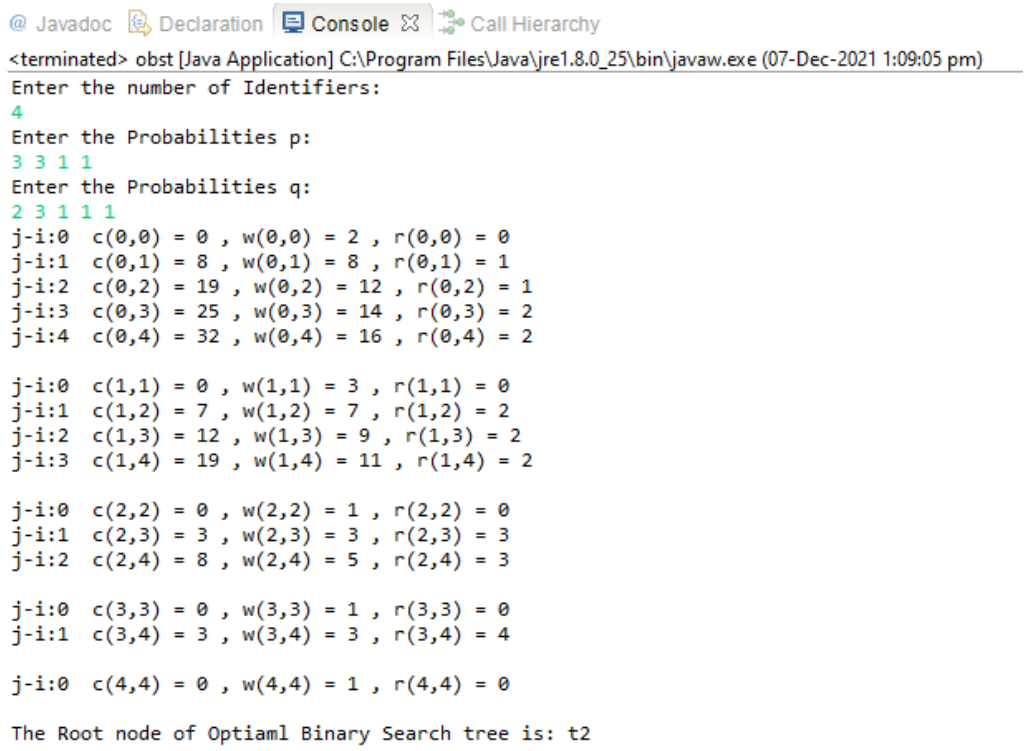
Aim: Write a java program to implement Dynamic Programming algorithm for the Optimal Binary Search Tree Problem

Program:

```
import java.util.Scanner;
public class obst
{
        static int find(int c[][],int i,int j){
  int min =1000,l=0;
                for(int k=i+1;k<=j;k++)                 {
                        if((c[i][k-1]+c[k][j]) < min)
                                {
                                min = c[i][k-1]+c[k][j];
                                l = k;
                                }
                }
        return l;
        }
public static void main(String[] args) {
System.out.println("Enter the number of Identifiers: ");
Scanner s = new Scanner(System.in);
int n = s.nextInt();
int p[] = new int[n+1];
int q[] = new int[n+1];
   System.out.println("Enter the Probabilities p: ");
   for(int i=1;i<=n;i++){
   p[i] = s.nextInt();
   }
   System.out.println("Enter the Probabilities q: ");
   int c[][] = new int[n+1][n+1];
   int w[][] = new int[n+1][n+1];
   int r[][] = new int[n+1][n+1];
   for(int i=0;i<=n;i++){
   q[i] = s.nextInt();
   w[i][i] = q[i];
   c[i][i] = 0;
   r[i][i] = 0;
   }
   for(int i=0;i<=n;i++)   {
            for(int j=1;j<=n;j++)          {
                    if(j!=i && j>i)
                    {
                    w[i][j] = p[j]+q[j]+w[i][j-1];
                    }
            }
   }
   for(int m =1;m<=n;m++)   {
            for(int i=0;i<n;i++)           {
                    for(int j=0;j<=n;j++)
                    {
                            if(j!=i && j>i && j-i==m)
                            {
                            int k = find(c,i,j);
                            r[i][j] = k;
```

```
                                c[i][j] = w[i][j]+c[i][k-1]+c[k][j];
                                }
                        }
                }
        }
        for(int i=0;i<=n;i++)
        {
                for(int j=0;j<=n;j++)
                {
                        if(j>=i)
                        {
                        System.out.print("j-i:");
                        System.out.print(+(j-i));
                        System.out.print("  ");
                        System.out.print("c("+i+","+j+") = "+c[i][j] +" , ");
                        System.out.print("w("+i+","+j+") = "+w[i][j] +" , ");
                        System.out.println("r("+i+","+j+") = "+r[i][j] +" ");
                        }
                 }
                System.out.println();
        }
        System.out.println("The Root node of Optiaml Binary Search tree is: t"+r[0][n]);

        s.close();
}
}
```

Output:



Aim: Write a Java program to implement Dynamic Programming algorithm for 0/1 Knapsack problem

Program:

```
import java.util.*;

public class knapsack01 {
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		System.out.println("Enter the no of instances:");
		int n = s.nextInt();
		int p[] = new int[n+1];
		int w[] = new int[n+1];
		   System.out.println("Enter the weights:");
		   for(int i=1;i<=n;i++){
		   w[i] = s.nextInt();
		   }
		   System.out.println("Enter the Profits:");
		   for(int i=1;i<=n;i++){
		   p[i] = s.nextInt();
		   }
		   System.out.println("Enter the maximum capacity");
		   int m=s.nextInt();
		   knapsack(w,p,m,n);
	}
	public static void knapsack(int W[],int P[],int M,int N)
	{
			int inf=Integer.MAX_VALUE;
			int[][] m=new int[N+1][M+1];
			int[][] s=new int[N+1][M+1];
			int i,j,n,w,tp=0,tw=0;
			for(i=1;i<=N;i++)
			{
				for(j=0;j<=M;j++)
				{
					int m1=m[i-1][j];
					int m2=inf;
					if(j>=W[i])
					{
						m2 = m[i - 1][j - W[i]] + P[i];
						m[i][j] = Math.max(m1, m2);
						s[i][j] = m2 > m1 ? 1 : 0;
					}
				}
			}
			int[] selected = new int[N + 1];
			for ( n = N, w = M; n > 0; n--)
			{
				if (s[n][w] != 0)
				{
					selected[n] = 1;
					tw+=W[n];
					tp+=P[n];
```

```
                    w = w - W[n];
                }
                else
                {
                    selected[n] = 0;
                }
            }
            System.out.print("Optimal sol is (");
            for( i=1;i<N+1;i++)
            {
                if(i!=N)
                    System.out.print("x"+i+",");
                else
                        System.out.print("x"+i+"): (");
            }
            for( i=1;i<N+1;i++)
            {
                if(i!=N)
                    System.out.print(selected[i]+",");
                else
                    System.out.println(selected[i]+")");
            }
            System.out.println("Total profit obtained : "+tp);
            System.out.println("Total weight obtained : "+tw);

        }


    }
```

Output:

@ Javadoc | Declaration | Console | Call Hierarchy

<terminated> knapsack01 [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.e

```
Enter the no of instances:
3
Enter the weights:
1 2 3
Enter the Profits:
2 5 6
Enter the maximum capacity
5
Optimal sol is (x1,x2,x3): (0,1,1)
Total profit obtained : 11
Total weight obtained : 5
```

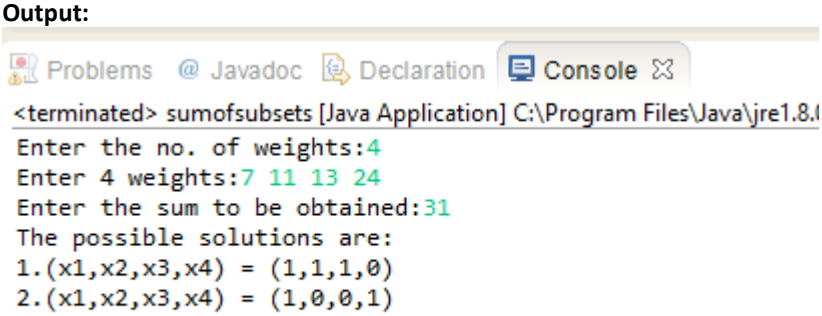
Aim: Write a java program to implement sum of subsets using backtracking Algorithm.

Program:

```
import java.util.*;
class sumofsubsets {
        static int[] w;
        static int[] x;
        static int m,n,c=0;
        public static void main(String args[])
                {
                        Scanner s=new Scanner(System.in);
                        System.out.print("Enter the no. of weights:");
                         n=s.nextInt();
                         w = new int[n + 1];
                         x= new int[n + 1];
                         int t= 0,i;
                         System.out.print("Enter " + n + " weights:");
                         for(i=1;i<n+1;i++)
                         {
                                 w[i]=s.nextInt();
                                 t+=w[i];
                         }
                         System.out.print("Enter the sum to be obtained:");
                         m=s.nextInt();
                         if (t<m || m<w[1])
                         {
                                 System.out.println("Not possible to obtain the subset!!");
                                 System.exit(1);
                         }
                         System.out.println("The possible solutions are:");
                         SumOfSub(0, 1, t);
                         //if(c==0)
                                 //System.out.println(c);



                }

        static public void SumOfSub(int s, int k, int r)
                {
                        int i=0,l;
                         x[k]=1;
                         if(s+w[k]==m)
                         {
                                         c++;
                                         System.out.print(""+c+".(");
                                         for(l=1;l<x.length;l++)
                                         {
                                                 if(l!=x.length-1)
                                                         System.out.print("x"+l+",");
                                                 else
                                                         System.out.print("x"+l);
                                        }
                                         System.out.print(") = (");
```

```
                                        for(i=1;i<=k;i++)
                                        {
                                                if(i!=k)
                                                        System.out.print(x[i]+",");
                                                else
                                                        System.out.print(x[i]);
                                        }
                                        for(i=k+1;i<x.length;i++)
                                                System.out.print(",0");
                                                        System.out.print(")\n");
                                }
                                else if((s+w[k]+w[k+1])<= m)
                                {
                                SumOfSub(s+w[k],k+1,r-w[k]);
                                }
                                if((s+r-w[k])>=m && (s+w[k+1])<=m)
                                {
                                x[k]=0;
                                SumOfSub(s,k+1,r-w[k]);
                                }
                        }

}
```

Output:



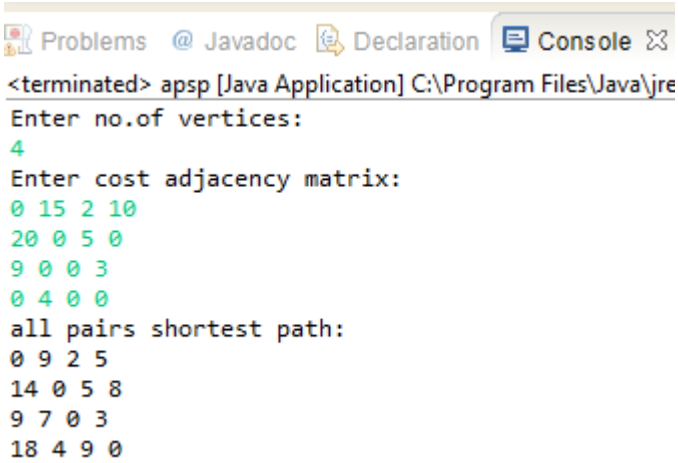
Aim:Write a java program to implement all pairs shortest path using Floyd triangle

Program:

```
import java.util.*;
public class apsp {
    public static void main(String args[])
    {
          Scanner sc=new Scanner(System.in);
                System.out.println("Enter no.of vertices: ");
                int v=sc.nextInt();
                int[][] cost=new int[v][v];
                int inf=9999;
                int i,j,k;
                System.out.println("Enter cost adjacency matrix: ");
                for(i=0;i<v;i++)
                {
                        for(j=0;j<v;j++)
                        {
                        cost[i][j]=sc.nextInt();
                        if(i!=j && cost[i][j]==0){
                                cost[i][j]=inf;
                        }
                        }
     }
          int cm[][]=new int[v][v] ;
          for( i = 0; i<v; i++)
          {
              for(j = 0; j<v; j++)
              {
                cm[i][j] = cost[i][j]; //copy costMatrix to new matrix
              }
          }
          for( k = 0; k<v; k++)
          {
                  for( i = 0; i<v; i++)
                  {
                    for( j = 0; j<v; j++)
                    {
                      if(cm[i][k]+cm[k][j] < cm[i][j])
                      {
                        cm[i][j] = Math.min(cm[i][j],cm[i][k]+cm[k][j]);
                      }
                    }
                  }
           }
          System.out.println("all pairs shortest path:");
          for(i=0;i<v;i++)
                  {
                        for(j=0;j<v;j++)
                        {
                                if(cm[i][j]>=9999)
                                {
                                        System.out.println("INF");
                                }
                                else
                                        System.out.print(cm[i][j]+" ");
                        }
```

```
                    System.out.println();
                }
    }
}
```

Output:



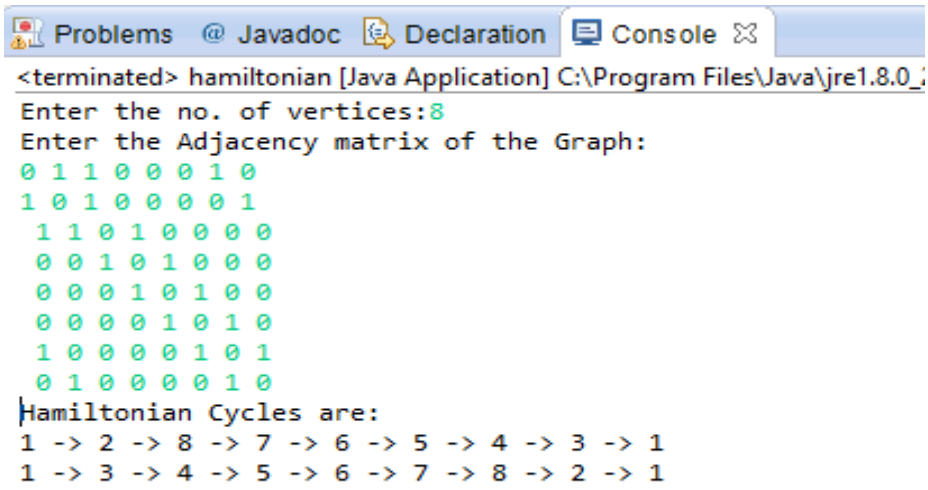
Aim: To write a java program to implement Backtracking algorithm for the Hamiltonian Cycles problem.

Program:

```
import java.util.*;
public class hamiltonian {
	static int n,c=0;
	 static int[] x;
	 static int[][] g;
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		 System.out.print("Enter the no. of vertices:");
		 n=s.nextInt();
		 int i,j,e,y,z;
		 x=new int[n+1];
		 x[1]=1;
		 for(i=2;i<=n;i++)
		 x[i]=0;
		 g=new int[n+1][n+1];
		 System.out.println("Enter the Adjacency matrix of the Graph:");
		 for(i=1;i<=n;i++)
		 {
			for(j=1;j<=n;j++)
				g[i][j]=s.nextInt();
		 }
		 System.out.println("Hamiltonian Cycles are:") ;
		 Hamiltonian(2);
		 if(c==0)
		 System.out.println(c+"\nNo Hamiltonian Cycle for the given Graph!!");
		}

	static void NextValue(int k)
	 {
		 int j;
		 do
		 {
			x[k]=(x[k]+1)%(n+1);
			if(x[k]==0)
				return;
			if(g[x[k-1]][x[k]]!=0)
			{
				for(j=1;j<k;j++)
				{
				if(x[j]==x[k])
				break;
				}
				if(j==k)
				{
				if((k<n) || ((k==n) && g[x[n]][x[1]]!=0))
				return;
				}
			}
		 }while(true);
	 }
	 static void Hamiltonian(int k)
```

```
        {
        int i;
        do{


              NextValue(k);
               if(x[k]==0)
                     return;
               if(k==n)
               {
                     for(i=1;i<=n;i++)
                     System.out.print(x[i]+" -> ");
                     System.out.print(x[1]);
                     System.out.println();
                     c++;
               }
               else
                     Hamiltonian(k+1);
        }while(true);
        }


}
```

Output:

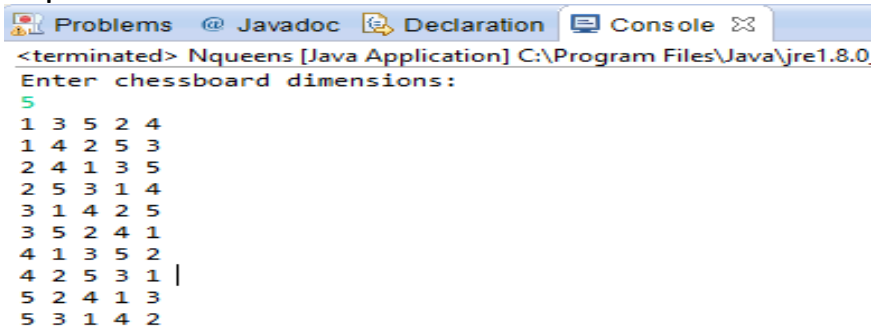


Aim: Write a Java program to implement Backtracking algorithm for the N-queens problem.

Program:

```
import java.util.*;
public class Nqueens{
	public boolean place(int k,int i,int x[]){
		int j;
		for(j=1;j<k;j++){
			if((x[j]==i || (Math.abs(x[j]-i)==Math.abs(j-k))))
			{
				return false;
			}
		}
		return true;
	}
		public void nqueens(int k,int n,int x[]){
			int i,j;
			for(i=1;i<=n;i++){
				if(place(k,i,x)==true){
					x[k]=i;
					if(k==n){
						for(j=1;j<=n;j++)
							System.out.print(x[j]+" ");
						System.out.println();
					}
					else
						nqueens(k+1,n,x);
				}
			}
		}
		public static void main(String args[])
		{
			int n;
			Scanner sc=new Scanner(System.in);
			System.out.println("Enter chessboard dimensions:");
			n=sc.nextInt();
			int x[]=new int[n+1];
			Nqueens obj=new Nqueens();
			obj.nqueens(1,n,x);
		}

	}
```

Output:



Aim: Write a Java program to implement Backtracking algorithm for the Sum of subsets problem.

Program:

```
import java.util.*;
class sumofsubsets {
        static int[] w;
        static int[] x;
        static int m,n,c=0;
        public static void main(String args[])
             {
                    Scanner s=new Scanner(System.in);
                    System.out.print("Enter the no. of weights:");
                     n=s.nextInt();
                     w = new int[n + 1];
                     x= new int[n + 1];
                     int t= 0,i;
                     System.out.print("Enter " + n + " weights:");
                     for(i=1;i<n+1;i++)
                     {
                            w[i]=s.nextInt();
                            t+=w[i];
                     }
                     System.out.print("Enter the sum to be obtained:");
                     m=s.nextInt();
                     if (t<m || m<w[1])
                     {
                            System.out.println("Not possible to obtain the
subset!!");
                            System.exit(1);
                     }
                     System.out.println("The possible solutions are:");
                     SumOfSub(0, 1, t);
                     //if(c==0)
                            //System.out.println(c);



             }

        static public void SumOfSub(int s, int k, int r)
             {
                    int i=0,l;
                     x[k]=1;
                     if(s+w[k]==m)
                     {
                                  c++;
                                  System.out.print(""+c+".(");
                                  for(l=1;l<x.length;l++)
                                  {
                                         if(l!=x.length-1)
                                                System.out.print("x"+l+",");
                                         else
                                                System.out.print("x"+l);
                                  }
                                  System.out.print(") = (");
                                  for(i=1;i<=k;i++)
```

```
                                        {
                                                if(i!=k)
                                                        System.out.print(x[i]+",");
                                                else
                                                        System.out.print(x[i]);
                                        }
                                        for(i=k+1;i<x.length;i++)
                                                System.out.print(",0");
                                                      System.out.print(")\n");
                        }
                        else if((s+w[k]+w[k+1])<= m)
                        {
                        SumOfSub(s+w[k],k+1,r-w[k]);
                        }
                        if((s+r-w[k])>=m && (s+w[k+1])<=m)
                        {
                        x[k]=0;
                        SumOfSub(s,k+1,r-w[k]);
                        }
                }

}
```

Output: