# Program 10

**10.A)**

Write java program to implement All-pairs Shortest paths problem using Floyd's algorithm

```
import java.util.Scanner;

public class P10 {

	public static void main(String[] args) {
		int i,j,k,n;
		int[][] a=new int[10][10];
		Scanner read=new Scanner(System.in);
		System.out.println("enter the no of nodes");
		n=read.nextInt();
		System.out.println("enter the cost adjancy matrix,'9999' for no direct path");
		for(i=1;i<=n;i++)
		{
			for(j=1;j<=n;j++)
				a[i][j]=read.nextInt();
			a[i][j]=0;
		}
		for(k=1;k<=n;k++)
			for(i=1;i<=n;i++)
				for(j=1;j<=n;j++)
					if(a[i][k]+a[k][j]<a[i][j])
						a[i][j]=a[i][k]+a[k][j];
		System.out.println("output path matrix");
		for(i=1;i<=n;i++)
		{
			for(j=1;j<=n;j++)
				System.out.print(a[i][j]+"\t");
			System.out.println();
		}


	}

}
```

**10.B)**

Implement Travelling sales person problem using Dynamic Programming

```
import java.util.Scanner;
public class P10b {
        static int [][] cost = new int [20][20];
        static int [] visited = new int [20];
        static int n,min_cost;
        static int Tsp_Dynamic(int i,int copy [])
        {
                int min=999,val,j;
                int [] s = new int [20];
                boolean flag=false;
                for(j=1;j<=n;j++)
                        s[j]=copy[j];
                s[i]=1;
                if(n==1)
                        return cost[i][1];
                for(j=1;j<=n;j++)
                {
                         if(s[j]==0)
                        {
                                flag=true;
                                val=cost[i][j]+Tsp_Dynamic(j,s);
                                 if(val<min)
                                        min=val;
                        }
                }
                  if(!flag)
                         min=cost[i][1];
                    return min;
        }
```

```
  public static void main(String[] args)
  {
        int i,j;
        Scanner read=new Scanner(System.in);
        System.out.println("Enter the number of  cities");
        n=read.nextInt();
        System.out.println("Enter the cost adjacency  matrix");
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
                cost[i][j]=read.nextInt();
        min_cost=Tsp_Dynamic(1,visited);
        System.out.println("The cost of optimal tour is "+ min_cost);


    }


}
```