1 a. Create a Java class called *Student* with the following details as variables within it.

(i) USN
(ii) Name
(iii) Branch
(iv) Phone

Write a Java program to create *nStudent* objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

```
//*************************** Student.java ************************

package daa.first.a;

public class Student {

	private  String  usn;
	private String name;
	private String branch;
	private String phoneNumber;

	public Student(String usn, String name, String branch, String
phoneNumber) {
		this.usn = usn;
		this.name = name;
		this.branch = branch;
		this.phoneNumber = phoneNumber;
	}

	@Override
	public String toString() {
		return "Student [usn=" + usn + ", name=" + name + ", branch=" +
branch
				+ ", phoneNumber=" + phoneNumber + "]";
	}
 }



//*************************** StudentDemo.java ************************

package daa.first.a;

import java.util.Scanner;

public class StudentDemo {
	public static void main(String[] args) {

		int i;
		String usn;
		String name;
		String branch;
		String phoneNumber;

		Scanner in = new Scanner(System.in);
		System.out.println("Enter number of Students");
		int n = in.nextInt();
```

```
            Student [] stu =new Student[10];

            for(i=0; i<n; i++){
                  System.out.println("Enter USN");
                  usn = in.next();

                  System.out.println("Enter Name");
                  name = in.next();

                  System.out.println("Enter Branch");
                  branch = in.next();

                  System.out.println("Enter Phone Number");
                  phoneNumber = in.next();

                  stu[i] = new Student(usn, name, branch, phoneNumber);
            }

            System.out.println("---------------Student Details -------------- ");

            for(i=0; i<n; i++){
                  System.out.println(stu[i]);
            }
      }
}
```

1.b Write a Java program to implement the Stack using arrays. Write Push(), Pop(), and Display() methods to demonstrate its working.

```
//*************************** Stack.java ************************

package daa.first.b;

public class Stack {

    private int top;
    private int [] arr;

    Stack(int size) {
        top = -1;
        arr = new int[size];
    }

    void push(int data) {
        if (top == arr.length - 1)
            System.out.println("Stack Overflow");
        else {
            arr[++top] = data;
            System.out.println("Pushed Item : " + arr[top]);
        }
    }

    void pop() {
        if (top < 0)
            System.out.println("Stack Underflow");
        else
            System.out.println("Poped Item : " + arr[top--]);
    }

    void display(){
      if (top < 0)
            System.out.println("Stack is Empty");
      else {
                System.out.println("Elements in the Stack  are: ");
                for(int i = top; i>=0; i--)
                      System.out.print(arr[i] + " ");
         }
    }
}
```

```java
//************************** StackDemo.java ************************

package daa.first.b;

import java.util.Scanner;

public class StackDemo {

	public static void main(String[] args) {
		Scanner in = new Scanner(System.in);
		System.out.println("Enter the Size of an Stack");
		int size = in.nextInt();
		Stack stk = new Stack(size);
	   boolean yes=true;
	   int choice;

	  while(yes){

	System.out.println("\n1.Push\n2.Pop\n3.Display\n4.Exit\n\nEnter Choice");
				choice = in.nextInt();

				switch(choice)
				{
				    case 1: System.out.println("Enter Push Item: ");
				            stk.push(in.nextInt());
				            break;
				    case 2: stk.pop();
				        break;
				    case 3: stk.display();
				         break;
				    case 4: yes = false;
				         break;
				    default: System.out.println("Invalid Choice");
				}
			}

		}

	}
```

Design a superclass called *Staff* with details as StaffId, Name, Phone, Salary. Extend this class by writing three subclasses namely *Teaching* (domain, publications), *Technical* (skills), and *Contract* (period). Write a Java program to read and display at least 3 *staff* objects of all three categories.

```
//*************************** Staff.java ************************

package daa.second.a;

public class Staff {
	protected String staffId;
	protected String name;
	protected long phNum;
	protected long salary;

	public Staff(String staffId, String name, long phNum, long salary) {
		this.staffId = staffId;
		this.name = name;
		this.phNum = phNum;
		this.salary = salary;
	}
 }

//*************************** Teaching.java ************************

package daa.second.a;

public class Teaching extends Staff {
	private String domain;
	private String publication;

	public Teaching(String staffId, String name, long phNum, long salary,
			String domain, String publication) {
		super(staffId, name, phNum, salary);
		this.domain = domain;
		this.publication = publication;
	}

	public String toString() {
		return "Teaching [domain=" + domain + ", publication=" + publication
				+ ", staffId=" + staffId + ", name=" + name + ", phNum="
				+ phNum + ", salary=" + salary + "]";
	}
 }
//*************************** Technical.java ************************

package daa.second.a;

public class Technical extends Staff {
	private String skills;
	public Technical(String staffId, String name, long phNum, long salary,
			String skills) {
		super(staffId, name, phNum, salary);
		this.skills = skills;
	}
```

```
	public String toString() {
		return "Technical [skills=" + skills + ", staffId=" + staffId
				+ ", name=" + name + ", phNum=" + phNum + ", salary=" +
salary
				+ "]";
	}
 }
//*************************** Contract.java ************************

package daa.second.a;

public class Contract extends Staff {
 private String period;

 public Contract(String staffId, String name, long phNum, long salary,
		String period) {
	super(staffId, name, phNum, salary);
	this.period = period;
   }

 public String toString() {
	return "Contract [Period=" + period + ", StaffId=" + staffId + ", Name="
			+ name + ", Phone Num=" + phNum + ", Salary=" + salary + "]";
	}
 }

//*************************** StaffDemo.java ************************

package nnn.cit;

import java.util.Scanner;

public class StaffDemo {

	public static void main(String[] args) {
		String staffId, name, domain, publication, skills, period;
		long salary, phNum;
		int teachingCount=0,technicalCount=0,contractCount=0;
		int i =0;

		Teaching[] teachingStaff = new Teaching[10];
		Technical[] technicalStaff = new Technical[10];
		Contract[] contractStaff = new Contract[10];

		Scanner in = new Scanner(System.in);
		while(true)
		 {
			System.out.println("Enter your choice");
			System.out.println("1 Teaching Staff Entry");
			System.out.println("2 Technical Staff Entry");
			System.out.println("3 Contract Staff Entry");
			System.out.println("4 Teaching Staff Details");
			System.out.println("5 Technical Staff Details");
```

```
                System.out.println("6 Contract Staff Details");
                System.out.println("7 Exit");

                int choice = in.nextInt();

                switch(choice)
                 {
                      case 1: System.out.println("Enter Teaching staff
Details(StaffId,Name,Salary,PhoneNumber,Domain,Publication)");
                                staffId = in.next();
                                name  =  in.next();
                                salary = in.nextLong();
                                phNum = in.nextLong();
                                domain = in.next();
                                publication = in.next();
                                teachingStaff[teachingCount]= new
Teaching(staffId,name,phNum,salary,domain,publication);
                                teachingCount++;
                                break;

                      case 2: System.out.println("Enter Technical
staffDetails(StaffId,Name,Salary,PhoneNumber,Skills)");
                                staffId = in.next();
                                name = in.next();
                                salary = in.nextLong();
                                phNum = in.nextLong();
                                skills = in.next();
                                technicalStaff[technicalCount] = new
Technical(staffId,name,phNum,salary,skills);
                                technicalCount++;
                                break;

                      case 3: System.out.println("enter Contract staff details
StaffId,Name,Salary,PhoneNumber,Period)");
                                staffId = in.next();
                                name = in.next();
                                salary = in.nextLong();
                                phNum = in.nextLong();
                                period = in.next();
                                contractStaff[contractCount] = new
Contract(staffId,name,phNum,salary,period);
                                contractCount++;
                                break;

                      case 4: System.out.println("Teaching Staff Details");
                                for(i=0;i<teachingCount;i++)
                                      System.out.println(teachingStaff[i]);
                                break;

                      case 5: System.out.println("Technical Staff Details:");
                                for(i=0;i<technicalCount;i++)
                                      System.out.println(technicalStaff[i]);
                                break;
```

```
                    case 6: System.out.println("contract Staff Details:");
                              for(i=0;i<contractCount;i++)
                                    System.out.println(contractStaff[i]);
                              break;

                    case 7: System.exit(0);

                }
            }
        }

    }

}
```

Write a Java class called *Customer* to store their name and date_of_birth. The date_of_birth format should be dd/mm/yyyy. Write methods to read customer data as <name, dd/mm/yyyy> and display as <name, dd, mm, yyyy> using StringTokenizer class considering the delimiter character as "/".

```
//*************************** Customer.java ************************

package daa.second.b;

import java.util.StringTokenizer;

public class Customer {
	private String customerName;
	private String date;

	public Customer(String customerName, String date) {
		this.customerName = customerName;
		this.date = date;
	}


	public String toString() {
		String returnVal = customerName + ", ";
		StringTokenizer st = new StringTokenizer(date,"/");
		System.out.println("Customer Details are");
		while(st.hasMoreTokens()){
			returnVal += st.nextToken();
			if(st.hasMoreElements())
				returnVal += "," ;
		}
		return returnVal;
	}
}
//*************************** CustomerDemo.java ************************

package daa.second.b;

import java.util.Scanner;

public class CustomerDemo {

	public static void main(String[] args) {
		String customerName;
		String date;
		Scanner in= new Scanner(System.in);
		System.out.println("Ënter customer name");
		customerName=in.next();
		System.out.println("Ënter Date (dd/mm/yyyy)");
		date=in.next();
		Customer ob = new Customer(customerName,date);
		System.out.println(ob);
	}

}
```

Write a Java program to read two integers *a* and*b*. Compute *a*/*b* and print, when *b* is not zero. Raise an exception when *b* is equal to zero.

```
//*************************** DivisionDemo.java ************************

package daa.thrid.a;

import java.util.Scanner;

public class DivisionDemo {

	public static void main(String[] args) {
		int a,b;
		int quotient;
		Scanner in = new Scanner(System.in);
		System.out.println("Please enter first number(numerator) : ");
		a = in.nextInt();
		System.out.println("Please enter second number(denominator) : ");
		b = in.nextInt();
		try{
			quotient = a / b;
			System.out.println ("Quotient = " + quotient);
		 }
		catch(ArithmeticException ae){
			System.out.println (ae);
		}
	}
}
```

Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer for every 1 second; second thread computes the square of the number andprints; third thread will print the value of cube of the number.

```
//************************** Square.java ************************

package daa.thrid.b;

public class Square implements Runnable {

	public int x;

	public Square(int x) {
		this.x = x;
	 }

	public void run() {
		System.out.println("Second Thread: Square of " + x + "is " + (x*x));
	 }
}

//************************** Cube.java ************************

package daa.thrid.b;

public class Cube implements Runnable{
	public int x;


	public Cube(int x) {
		this.x = x;
	}

	public void run() {
		System.out.println("Thrid Thread: Cube of " + x + "is " + (x*x*x));


	}

}

//************************** FirstThread.java ************************

package daa.thrid.b;

import java.util.Random;

public class FirstThread implements Runnable {

		public void run(){
			int num = 0;
			Random r = new Random();
```

```
                try
                {
                      for (int i = 0; i < 5; i++)
                       {
                            num = r.nextInt(100);
                            System.out.println("Main Thread Started and
Generated Number is " + num);
                            Thread t2 = new Thread(new Square(num));
                            t2.start();
                            Thread t3 = new Thread(new Cube(num));
                            t3.start();
                            Thread.sleep(1000);
                            System.out.println("----------------------");
                       }
                }
                catch (Exception ex){
                      System.out.println(ex.getMessage());
                }
          }
 }

//************************** MultiThreadDemo.java ************************

package daa.thrid.b;

public class MultiThreadDemo {

      public static void main(String[] args) {
            FirstThread ob = new FirstThread();
            Thread t = new Thread(ob);
            t.start();
      }
}
```

4. Sort a given set of *n* integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of *n*> 5000 and record the time taken to sort. Plot a graph of the time taken versus *n*on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide-and-conquer method works along with its time complexity analysis: worst case, average case and best case.

```
//************************** QuickSort.java ************************

package daa.fourth;

import java.util.Scanner;

public class QuickSort {
	int arr[];
	int n;
	Scanner ip = new Scanner(System.in);
	final int INFINITY = 999;

	void read(){
		int i;
		System.out.println("Enter the value of n");
		n= ip.nextInt();
		System.out.println("Enter the " + n +" Values");
		a = new int[n];
		for(i=0; i<n; i++){
			arr[i] = ip.nextInt();
		}
		arr[i] = INFINITY;
	}


	void display(){
		for(int i=0; i<n; i++){
			System.out.println(arr[i]+ " ");
		}
	}


	void quick(int low, int high){
		int j;
		if(low<high){
			j=partition(low, high);
			quick(low, j-1);
			quick(j+1,high);
		}
	}
```

```
	int partition(int low, int high){
		int key, temp, i, j;
		key = A[low];
		i=low+1;
		j = high;
		while(true){
			while(A[i] < key) i+=1;
			while(A[j] > key) j-=1;
			if(i<j){
				temp=A[i];
				A[i]=A[j];
				A[j]=temp;
			}
			else{
				A[low] = A[j];
				A[j] = key;
				return j;
			}
		}
	}
}

//***************************** QuickDemo.java ******************************

package daa.fourth;

public class QuickDemo {
	public static void main(String[] args) {

		QuickSort obj = new QuickSort();

		obj.read();
		System.out.println("Given Array is: ");
		obj.display();

		long start_time = System.nanoTime();
		obj.quick(0, obj.n-1);
		long end_time = System.nanoTime();

		System.out.println("Sorted Array is: ");
		obj.display();

		double elapsed_time = end_time - start_time;
		System.out.println("Elasped time is: " + elapsed_time + " ns");
	}
}
```

5. Sort a given set of *n* integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of *n*> 5000, and record the time taken to sort. Plot a graph of the time taken versus *n*on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide-and-conquer method works along with its time complexity analysis: worst case, average case and best case.

```
//*************************** MergeSortAlgm.java ****************************

package daa.fifth;

import java.util.Scanner;

public class MergeSortAlgm {
	int arr[];
	int n;
	Scanner ip = new Scanner(System.in);
	final int INFINITY = 999;

	void read(){
		int i;
		System.out.println("Enter the value of n");
		n= ip.nextInt();
		System.out.println("Enter the " + n +" Values");
		arr = new int[n];
		for(i=0; i<n; i++){
			arr[i] = ip.nextInt();
		}
		arr[i] = INFINITY;
	}


	void display(){
		for(int i=0; i<n; i++){
			System.out.println(arr[i]+ " ");
		}
	}


	void mergeSort(int low, int high){
		int j, mid;
		if(low<high){
			mid = (low+high)/2;
			mergeSort(low, mid);
			mergeSort(mid+1,high);
			merge(low, mid, high);
		}
	}
```

```
	void merge(int low, int mid, int high){
		int h = low, i= low, j = mid+1;
		int B[] = new int[20];
		while(h <= mid && j <= high){
			if (arr[h] <= arr[j]){
				B[i] = arr[h];
				h++;
			}
			else {
				B[i] = arr[j];
				j++;
			}
			i++;
		}
		if(h>mid){
			for(int k = j; k<=high; k++){
				B[i] = arr[k];
				i++;
			}
		}
		else{
			for(int k=h; k<=mid; k++){
				B[i] = arr[k];
				i++;
			}
		}
		for(int k=low; k<=high; k++)
			arr[k] = B[k];
	}
}


//**************************** MergeDemo.java ****************************
package daa.fifth;

public class MergeDemo {
	public static void main(String[] args) {
		MergeSortAlgo obj = new MergeSortAlgo();
		obj.read();

		System.out.println("Given Array is: ");
		obj.display();

		long start_time = System.nanoTime();
		obj.mergeSort(0, obj.n-1);
		long end_time = System.nanoTime();

		System.out.println("Sorted Array is: ");
		obj.display();

		double elapsed_time = end_time - start_time;
		System.out.println("Elasped time is: " + elapsed_time + " ns");
	}
}
```

6. Implement in Java, the 0/1 Knapsack problem using (a) Dynamic Programming method (b) Greedy method.

(a) Dynamic Programming

```
//*************************** KnapSack.java ******************************

package daa.sixth.a;

import java.util.Scanner;

public class KnapSack {
	int n, p[], w[], cap;
	Scanner ip = new Scanner(System.in);
	public KnapSack() {
		w = new int[10];
		p = new int[10];
	}

	void read(){
		int i;
		System.out.println("Enter the no of objects");
		n = ip.nextInt();
		System.out.println("Enter " + n+ " weights and profits: ");
		for(i=1;i<=n;i++){
			w[i] = ip.nextInt();
			p[i] = ip.nextInt();
		}
		System.out.println("Enter the capacity of the sack");
		cap = ip.nextInt();
	}

	int dynamicKnapSack(int i, int j){
		if (i==0 || j==0)
			return 0;
		if (j<w[i])
			return dynamicKnapSack(i-1, j);
		else
			return(Math.max(dynamicKnapSack(i-1,j),p[i]+dynamicKnapSack(i-1, j-w[i])));
	}

	public static void main(String args[]){
		int profit;
		KnapSack obj = new KnapSack();
		obj.read();
		profit = obj.dynamicKnapSack(obj.n, obj.cap);
		System.out.println("Optimal Solution is: " + profit);
	 }
}
```

(b) Greedy Method

```
//*************************** KnapSackGreedy.java ***************************

   package daa.sixth.b;

   import java.util.Scanner;

   public class KnapSackGreedy {
      int n, p[], w[], cap;
      float ratio[];
      Scanner ip = new Scanner(System.in);
      public KnapSackGreedy() {
            w = new int[20];
            p = new int[20];
            ratio = new float[20];
      }

      void read(){
            int i;
            System.out.println("Enter the no of objects");
            n = ip.nextInt();
            System.out.println("Enter " + n+ " weights and profits: ");
            for(i=1;i<=n;i++){
                  w[i] = ip.nextInt();
                  p[i] = ip.nextInt();
                  ratio[i] = p[i] / w[i];
            }
            System.out.println("Enter the capacity of the sack");
            cap = ip.nextInt();
      }

      int largest(){
            int i, index = 1;
            float big = ratio[1];
            for(i=2; i<=n; i++){
                  if(ratio[i] > big){
                        big = ratio[i];
                        index = i;
                  }
            }
            ratio[index]= 0;
            return index;
      }

      void gKnapSack(int m, int n){
            float x[] = new float [20];
            int i, index = 0;
            float optimalSol = 0.0f;
            for(i=1; i<=n; i++){
                  x[i] = 0;
            }
            int u = m;
```

```
        for (i = 1; i<=n; i++){
              index = largest();
              if(w[index] > u) break;
              x[index] = 1;
              u = u-w[index];
        }
        if(index<=n){
              x[index] = ((float)u/w[index]);
        }
        for(i=1; i<=n; i++){
              if(x[i] != 0.0){
                    optimalSol = optimalSol+ (x[i] *p[i]);
              }
        }
        System.out.println("Optimal Solution is: "+ optimalSol);
    }


    public static void main(String args[]){
        KnapSackGreedy obj = new KnapSackGreedy();
        obj.read();
        obj.gKnapSack(obj.cap, obj.n);
     }
}
```

7. From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. Write the program in Java.

```
//*************************** DijkstraGraph.java ****************************

package daa.seventh;

import java.util.Scanner;

public class DijkstraGraph {
	int cost[][], dist[], s[], n, u, startingNode;

	void readGraph(){
		int i, j;
		Scanner ip = new Scanner(System.in);
		cost=new int[10][10];
		dist= new int [10];
		s = new int [10];
		System.out.println("Enter the number of vertices");
		n = ip.nextInt();
		System.out.println("Enter the cost Adjacent matrix: ");

		for(i=1; i<=n; i++){
			for(j=1; j<=n; j++){
				cost[i][j] = ip.nextInt();
			}
		}
		System.out.print("Enter the starting Vertex:");
		startingNode = ip.nextInt();
	}

	void ssp(){
		int i, min;
		for(i=1; i<=n; i++){
			s[i] = 0;
			dist[i] = cost[startingNode][i];
		}
		s[startingNode] = 1;
		dist[startingNode] = 0;
		for(int num = 2; num <= n-1; num++){
			min = 999;
			for(int w = 1; w<=n; w++){
				if(s[w] == 0 && dist[w] < min){
					min = dist[w];
					u = w;
				}
			}
			s[u] = 1;
			for(int w=1; w<=n; w++){
				if(s[w] == 0)
					dist[w] = Math.min(dist[w], dist[u] + cost[u][w]);
			}
		}
	}
```

```
	void display(){
		for(int i=1; i<=n; i++){
			System.out.println("Shortest path from node "+ startingNode +
" to " + i + " --> "+ dist[i]);
		}
	}
}


//**************************** DijkstraDemo.java ****************************

package daa.seventh;

public class DijkstraDemo {
	public static void main(String args[]){
		DijkstraGraph obj = new DijkstraGraph();
		obj.readGraph();
		obj.ssp();
		obj.display();
	}
}
```

8. Find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm. Use Union-Find algorithms in your program.

```
//***************************** KruskalGraph.java ******************************

package daa.eighth;

import java.util.Scanner;

public class KruskalGraph {
	int cost[][], s[], n;
	Scanner ip=new Scanner(System.in);

	void ReadGraph(){
		System.out.println("Enter the number of nodes");
		n=ip.nextInt();
		cost= new int[10][10];
		System.out.println("Enter the cost matrix");
		for(int i=1; i<=n; i++)
			for (int j=1; j<=n; j++ )
				cost[i][j]=ip.nextInt();
	}

	void kruskal(){
		int u = 0, v = 0, min_cost=0;
		s =new int[10];
		for(int i=1; i<=n;i++)
			s[i] = -1;
		System.out.println("Spanning Tree");
		System.out.println("Edges --> Cost");
		int num = 1;
		while(num<= n-1){
			int min = 999;
			for(int i=1; i<=n; i++){
				for(int j=1; j<=n; j++){
					if(cost[i][j] !=0 && cost[i][j] <min){
						u = i;
						v = j;
						min = cost[i][j];
					}
				}
			}
			int a = find(u);
			int b = find(v);
			System.out.print("(" +u+","+v+") -->");
			if(a != b){
				System.out.println(cost[u][v]);
				min_cost = min_cost+cost[u][v];
				union(u,v);
				num++;
			}
			else{
				System.out.println("Reject!! Because it forms cycle");
			}
```

```
                cost[u][v] = cost[v][u] = 999;
            }

            System.out.println("Minimum cost of Spanning Tree " + min_cost);
        }


        int find(int i){
            while(s[i] != -1)
                i = s[i];
            return i;
        }

            void union(int i, int j){
            s[find(j)] = i;
        }
    }

//**************************** KruskalDemo.java *****************************

package daa.eighth;

public class KruskalMain {

        public static void main(String[] args) {
            KruskalGraph obj = new KruskalGraph();
            obj.ReadGraph();
            obj.kruskal();
        }
    }
```

9. Find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

```
//***************************** PrimsGraph.java ****************************

package daa.ninth;

import java.util.Scanner;

public class PrimsGraph {
	int cost[][], span_tree[],n;

	void readGraph(){
		Scanner ip = new Scanner(System.in);
		System.out.println("Enter the number of nodes");
		cost= new int[10][10];
		n=ip.nextInt();
		System.out.println("Enter the cost matrix");
		for(int i=1; i<=n; i++)
			for (int j=1; j<=n; j++ )
				cost[i][j]=ip.nextInt();
	}

	void prims(){
		int u = 0, v = 0, min_cost=0;
		span_tree = new int[10];
		for(int i=1; i<=n; i++)
			span_tree[i] = 0;
		System.out.println("Edges --> Cost");
		span_tree[1] = 1;
		for (int num = 1; num<=n-1; num++){
			int min = 999;
			for(int i=1; i<= n; i++)
				for(int j=1; j<=n; j++)
					if(span_tree[i] == 1 && span_tree[j] == 0 && cost[i][j] <min){
						u = i;
						v = j;
						min = cost[i][j];
					}
			span_tree[v] = 1;
			System.out.println("(" +u+","+v+") -->" + cost[u][v]);
			min_cost += cost[u][v];
		}
		System.out.println("Minimum Cost of spanning Tree " + min_cost);
	}
}
```

```java
//*************************** PrimsDemo.java ****************************

package daa.ninth;

public class PrimsDemo {

	public static void main(String[] args) {
		PrimsGraph obj = new PrimGraph();
		obj.readGraph();
		obj.prims();
	}
}
```

10. Write Java programs to
(a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm.
(b) Implement Travelling Sales Person problem using Dynamic programming.

(a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm.

```
//*************************** Floyd.java *****************************

package daa.tenth.a;

public class Floyd {
	int [][] cost;
	int n;

	public Floyd(int[][] cost, int n) {
		super();
		this.cost = cost;
		this.n = n;
	}

	public void findPath(){
		int i,j,k;
		for(i=0; i<n; i++){
			for(j=0; j<n; j++){
				for(k=0; k<n; k++){
					if(cost[i][j] > cost[i][k] + cost[k][j])
						cost[i][j] =  cost[i][k] + cost[k][j];
				}
			}
		}

		System.out.println("The Shortest Path Obtained after applying
Floyd's Algorithm");
		for(i=0; i<n; i++){
			for(j=0; j<n; j++){
				System.out.print(cost[i][j] + " ");
			}
			System.out.println();
		}
	}
 }
```

```java
//*************************** FloydDemo.java ****************************

package daa.tenth.a;

import java.util.Scanner;

public class FloydDemo {

	public static void main(String[] args) {
		int [][] adj;
		int n;

		Scanner ip = new Scanner(System.in);
		System.out.println("Enter Number of Vertices");
		n = ip.nextInt();

		adj = new int[n][n];
		System.out.println("Enter the Adjacent Matrix Values");
		for(int i = 0; i<n; i++){
			for(int j = 0; j<n; j++){
				adj[i][j] = ip.nextInt();
			}
		}

		ip.close();

		Floyd ob = new Floyd(adj,n);
		ob.findPath();
	}
}
```

(b) Implement Travelling Sales Person problem using Dynamic programming.

```
//**************************** TSP.java ****************************

package daa.tenth.b;

import java.util.Scanner;

public class TSP {
	int costMat[][], n, cost, visited[];

	public TSP() {
		costMat = new int[10][10];
		visited = new int[10];
		cost = 0;
	}

	void read(){
		Scanner ip = new Scanner(System.in);
		System.out.println("Enter the number of nodes in graph");
	   n=ip.nextInt();
		System.out.println("Enter the adjacency matrix");
	   for(int i=0;i<n;i++)
		for(int j=0;j<n;j++)
			costMat[i][j]=ip.nextInt();
	   for(int i=0; i<n;i++)
		visited[i] = 0;
	}

	void min_cost(int city){
		int i, nCity;
		visited[city] = 1;
		System.out.println(" "+ (city+1));
		nCity = least(city);
		if (nCity == 999){
			nCity = 0;
			System.out.println(" "+ (nCity+1));
			cost += costMat[city][nCity];
			return;
		}
		min_cost(nCity);
	}
```

```
	int least(int c){
		int i, nc = 999;
		int min=999, kmin=0;
		for(i=0;i<n;i++){
			if((costMat[c][i] != 0) && (visited[i] == 0)){
				if(costMat[c][i] < min){
					min = costMat[i][0] + costMat[c][i];
					kmin = costMat[c][i];
					nc = i;
				}
			}
		}
		if(min != 999)
			cost += kmin;
		return nc;
	}

	void display(){
		System.out.println("min cost = " + cost);
	}
}

//***************************** TspDemo.java ******************************

package daa.tenth.a;

public class TspDemo {
	public static void main(String args[]){
		TSP obj = new TSP();
		obj.read();
		System.out.println("Path is: ");
		obj.min_cost(0);
		obj.display();
	}
}
```

11. Design and implement in Java to find a subset of a given set S = {Sl, S2,.....,Sn} of *n* positive integers whose SUM is equal to a given positive integer *d*. For example, if S ={1, 2, 5, 6, 8} and *d*= 9, there are two solutions {1,2,6}and {1,8}. Display a suitable message, if the given problem instance doesn't have a solution.

```
//***************************** Subset.java ******************************

package daa.eleventh;

public class Subset {
	int eleInSet[], subset[], sum, count=0;

	Subset(){
		eleInSet = new int[20];
		subset = new int[20];
	}

	void subsetSum(int k, int subSum, int totalSum){
		eleInSet[k]=1;
		if(subset[k]+subSum==sum){
			count++;
			System.out.println();
			for(int i=1;i<=k;i++)
				if(eleInSet[i] == 1)
					System.out.println(subset[i]+" ");
		}
		else
		if(subset[k] + subset[k+1] + subSum <= sum)
			subsetSum(k+1, subSum + subset[k], totalSum - subset[k]);
		if(subSum+totalSum-subset[k] >= sum && subSum + subset[k] <= sum){
			eleInSet[k]=0;
			subsetSum(k+1, subSum, totalSum-subset[k]);
		}
	}
 }

//***************************** SubsetDemo.java ******************************

package daa.eleventh;

import java.util.Scanner;

public class SubsetDemo {
	public static void main(String agrs[]){
		int n, totalSum=0;
		Scanner ip = new Scanner(System.in);
		Subset obj = new Subset();

		System.out.println("enter the number of elements:");
		n = ip.nextInt();
		System.out.println("enter the array elements:");
		for(int i=1;i<=n;i++){
			obj.subset[i] = ip.nextInt();
			totalSum += obj.subset[i];
```

```
            }
            System.out.println("enter sum value:");
            obj.sum = ip.nextInt();

            obj.subsetSum(1,0,totalSum);
            if(obj.count==0)
                  System.out.println("solution does not exists");
            else{
                  System.out.println("number of subsets are:"+ obj.count);
            }
      }
}
```

12. Design and implement in Java to find all Hamiltonian Cycle in a connected undirected graph G of n vertices using backtracking principle.

```
//************************HamiltonianCycle.java ****************************

package daa.tenth.a;

import java.util.Scanner;

class HamiltonianCycle
{
       int n;
       int path[];
       int graph[][];

       public HamiltonianCycle() {
            path =  new  int[10];
            graph = new int [10][10];
       }

 void read(){
       Scanner s =new Scanner(System.in);
     System.out.println("Enter the number of nodes in graph");
     n=s.nextInt();
     System.out.println("Enter the adjacency matrix");
     for(int i=1;i<=n;i++)
      for(int j=1;j<=n;j++)
            graph[i][j]=s.nextInt();
 }

boolean isSafe(int v, int path[], int pos)
 {
     if (graph[path[pos - 1]][v] == 0)
         return false;

     for (int i = 1; i <= pos; i++)
         if (path[i] == v)
             return false;

     return true;
 }

 boolean hamCycleUtil(int path[], int pos)
 {
     if (pos == n+1)
     {
         if (graph[path[pos - 1]][path[1]] == 1)
             return true;
         else
             return false;
     }

     for (int v = 2; v <= n; v++)
     {
```

```
            if (isSafe(v, path, pos))
            {
                path[pos] = v;
                if (hamCycleUtil(path, pos + 1) == true)
                    return true;
                path[pos] = -1;
            }
        }

        return false;
    }

    int hamCycle()
    {
       // path = new int[n];
        for (int i = 1; i <= n; i++)
            path[i] = -1;

        path[1] = 1;
        if (hamCycleUtil(path, 2) == false)
        {
            System.out.println("\nSolution does not exist");
            return 0;
        }

        printSolution(path);
        return 1;
    }

         void printSolution(int path[]){
              System.out.println("Solution Exists: Following is one Hamiltonian Cycle");
              for (int i = 1; i <= n; i++)
                   System.out.print(" " + path[i] + " ");

              System.out.println(" " + path[1] + " ");
         }

          public static void main(String args[]){
              HamiltonianCycle hamiltonian = new HamiltonianCycle();
              hamiltonian.read();
              hamiltonian.hamCycle();
         }
    }
```