Sahyadri College of Engineering and Management
Adyar, Mangaluru - 575007, Karnataka

Department of Information Science & Engineering

LABORATORY MANUAL

NAME OF THE COURSE	:	**DESIGN AND ANALYSIS OF ALGORITHM LABORATORY**
COURSE CODE	:	18CSL47
NBA COURSE CODE	:	C217
NAME OF THE MODULE	:	
NAME OF THE FACULTY INCHARGE	:	Mrs. Akhial Thejaswi R/Mrs. Suchetha G
NAME OF THE INSTRUCTOR INCHARGE	:	Mrs. Swayamprabha
ACADEMIC YEAR	:	2019 - 2020
SEMESTER	:	IV

TECHNICAL DETAILS

LAB NAME	:	ISE LAB –2
TOTAL NO. OF COMPUTERS	:	27
BRAND OF COMPUTER	:	ACER
SPECIFICATION	:	INTEL CORE™i3@3.40.GHz
		4 GB DDR3 RAM , 665 MHZ
		500 GB SATA HARD DISK
		19’’ TFT MONITOR
OPERATING SYSTEM	:	UBUNTU, WINDOWS-7
SOFTWARE	:	ORACLE 10G ,**NETBEANS 8.2**

DO’S AND DON’TS IN THE LABORATORY

1. Remove footwear and keep it in the rack before entering the lab.
2. Make entry in the Log Book as soon as you enter the laboratory.
3. All the students should sit according to their roll numbers starting from their left to right.
4. All the students are supposed to enter the terminal number in the log book.
5. Do not change the terminal on which you are working.
6. All the students are expected to get at least the concept to be implemented.
7. Strictly observe the instructions given by the Faculty/Lab Instructor.
8. The record should have following details:
    a) Date
    b) Aim
    c) Program
    d) Output

DESIGN AND ANALYSIS OF ALGORITHM LABORATORY

17CSL47

Experiments			**Page No**
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 *n Student* objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.	
	B	Write a Java program to implement the Stack using arrays. Write Push(), Pop(), and Display() methods to demonstrate its working.	
2	**A**	Design a super class 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.	
	B	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 “/”.	
3	**A**	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.	
	B	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 and prints; third thread will print the value of cube of the number.	
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.		
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.	
6	Implement in Java, the **0/1 Knapsack** problem using (a) Dynamic Programming method (b) Greedy method.	
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.	
8	Find Minimum Cost Spanning Tree of a given connected undirected graph using **Kruskal's algorithm.** Use Union-Find algorithms in your program.	
9	Find Minimum Cost Spanning Tree of a given connected undirected graph using **Prim's algorithm**.	
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.	
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.	
12	Design and implement in Java to find all **Hamiltonian Cycles** in a connected undirected Graph G of *n* vertices using backtracking principle.	

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 n Student objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

```
package javaapplication1;
import java.util.Scanner;
public class student
{
        String USN;
        String Name;
        String branch;
        Long phone;
        void insertRecord(String reg,String name, String brnch,Long ph)
        {
                USN=reg;
                Name=name;
                branch=brnch;
                phone=ph;
        }
        void displayRecord()
        {
                System.out.println(USN+" "+Name+" "+branch+" "+phone);
        }
        public static void main(String args[])
        {
                student s[]=new student [100];
                Scanner  sc=new Scanner(System.in);
                System.out.println("Enter the number of students");
                int n=sc.nextInt();
                for(int i=0;i<n;i++)
                s[i]=new student();
                for(int j=0;j<n;j++)
                {
                                System.out.println("Enter the usn,name,branch,phone");
                                String USN=sc.next();
                                String Name=sc.next();
                                String branch=sc.next();
                                Long  phone=sc.nextLong();
                        s[j].insertRecord(USN,Name,branch,phone);
                }
                for( int m=0;m<n;m++)
                {
                        System.out.println("USN\tNAME\tBRANCH\tPHONE");
                                s[m].displayRecord();
                }
        }
}
```

OUTPUT:

```
Enter the number of students
3
Enter the usn,name,branch,phone
101
Shanvi
ISE
9449389267
Enter the usn,name,branch,phone
102
Manvi
CSE
9739490048
```

```
Enter the usn,name,branch,phone
103
Gyanvi
ECE
9164874890

USN         NAME        BRANCH          PHONE
101         Shanvi          ISE         9449389267
102         Manvi           CSE         9739490048
103         Gyanvi          ECE         9164874890
```

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

```
package javaapplication2;
import java.util.Scanner;
public class stack
{
	final int max=5;
	int s[]=new int[max];
	int top=-1;
	void push(int ele)
	{
		if(top>=max-1)
		System.out.println("stack overflow");
		else
		s[++top]=ele;
	}
```

```
int pop()
{
        int z=0;
        if(top==-1)
        System.out.println("stack underflow");
        else
        z=s[top--];
        return z;
}
void display()
{
        if(top==-1)
        System.out.println("stack empty");
        else
        {       for(int i=top;i>-1;i--)
                System.out.println(s[i]+" ");
        }
}
public static void main(String args[])
{
        int q=1;
        stack m = new stack();
        System.out.println("program to perform stack operations");
            Scanner sc=new Scanner(System.in);
        while(q!=0)
        {
                System.out.println("enter 1. push 2.pop 3. display ");
                                System.out.println("enter your choice");
                int ch=sc.nextInt(); switch(ch)
                {
```

```
                              case 1:System.out.println("enter the element to be pushed");
                                               int ele=sc.nextInt();
                               m.push(ele);
                                        break;
                              case 2:int popele;
                              popele=m.pop();
                                   System.out.println("the poped element is");
                              System.out.println(popele+" ");
                                        break;
                              case 3:System.out.println("elements in the stack are");
                                        m.display();    break;
                              case 4:
                              q=0;
                         }
                    }
          }
```

OUTPUT:

```
program to perform stack operations
enter 1. push 2.pop 3. display
enter your choice
1
enter the element to be pushed
10
enter 1. push 2.pop 3. display
enter your choice
1
enter the element to be pushed
20
enter 1. push 2.pop 3. display
enter your choice
1
```

```
enter the element to be pushed
30
enter 1. push 2.pop 3. display
enter your choice
1
enter the element to be pushed
40
enter 1. push 2.pop 3. display
enter your choice
1
enter the element to be pushed
50
enter 1. push 2.pop 3. display
enter your choice
1
enter the element to be pushed
60
stack overflow
enter 1. push 2.pop 3. display
enter your choice
3
elements in the stack are
50
40
30
20
10
```

```
enter 1. push 2.pop 3. display
enter your choice
2
the poped element is
50
enter 1. push 2.pop 3. display
enter your choice
2
the poped element is
40
enter 1. push 2.pop 3. display
enter your choice
3
elements in the stack are
30
20
10
enter 1. push 2.pop 3. display
enter your choice
2
the poped element is
30
enter 1. push 2.pop 3. display
enter your choice
the poped element is
20
enter 1. push 2.pop 3. display
enter your choice
2
2
the poped element is
10
```

```
enter 1. push 2.pop 3. display


enter your choice
2
stack underflow
the poped element is
0
enter 1. push 2.pop 3. display
enter your choice
2
stack underflow
```

2a. Design a super class 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

```
package javaapplication3;
import java.util.Scanner;
public class Staff
{
	int staffid, String name;
	String phone;
	int salary;
	Scanner input=new Scanner(System.in);
	Scanner in=new Scanner(System.in);
	void read()
	{
		System.out.println("enter the staffid");
		staffid=input.nextInt();
		System.out.println("enter the name");
		name=in.nextLine();
		System.out.println("enter the phone");
		phone=in.nextLine();
		System.out.println("enter the salary");
		salary=input.nextInt();
	}
}
class teaching extends Staff
{
	String domain;
	String publisher;
	void read()
	{
```

```
                super.read();
                System.out.println("enter the domain");
                domain=in.nextLine();
                System.out.println("enter the publication");
                publisher=in.nextLine();
        }
        void display()
        {
System.out.println(staffid+"\t"+name+"\t"+phone+"\t"+salary+"\t"+domain+"\t"+publisher+"\t");
        }
}
class technical extends Staff
{
        int n;
        String[]skills;

        void read()
        {
                super.read();
                        System.out.println("enter the number of skills");
                        n=input.nextInt();
                        skills=new String[n];
                        for(int i=0;i<n;i++)
                        {
                        System.out.println("enter skills" +(i+1));
                        skills[i]=input.next();
                        }
        }
        void display()
        {
                        System.out.println(staffid+"\t"+name+"\t"+phone+"\t"+salary+"\t");
                for(int j=0;j<skills.length;j++)
```

```
                        {
                        System.out.print(skills[j]+",");
                        }
                        System.out.print("\n");
            }
 }
 class contractor extends Staff
 {
            int period;
            void read()
            {
                        super.read();
                        System.out.println("how many years of contract");
                        period=input.nextInt();
            }
            void display()
            {
                        System.out.println(staffid+"\t"+name+"\t"+phone+"\t"+salary+"\t"+period+"\t");
            }
}
class MYAPP
{
            static int n,m,o;
            static Scanner input=new Scanner(System.in);
            public static void main(String[] args)
            {
                        teaching[]s;
                        technical[]t;
                        contractor[]c;
                        System.out.println("\n******read teaching staff detail******");

                        System.out.println("how many number of teaching staff:");
```

```
m=input.nextInt();
s=new teaching[m];
for(int i=0;i<m;i++)
{
        s[i]=new teaching();
    System.out.println("enter teaching staff"+(i+1)+"details");
        s[i].read();
}
    System.out.println("\n******read technical staff detail******");
System.out.println("how many number of technical staff:");
n=input.nextInt();
t=new technical[n];
for(int i=0;i<n;i++)
{
        t[i]=new technical();
        System.out.println("enter technical staff"+(i+1)+"details");
        t[i].read();
}
System.out.println("\n******read contractor staff detail******");
System.out.println("how many number of contractor staff:");
o=input.nextInt();
c=new contractor[o];
for(int i=0;i<o;i++)
{
        c[i]=new contractor();
        System.out.println("enter contractor staff"+(i+1)+"details");
        c[i].read();
}
System.out.println("\n *****TEACHING STAFF *****");
System.out.println("SID\tNAME\tPHONE\\ttSALARY\tDOMAIN\tPUBLICATION");
    System.out.println("------");
    for(int i=0;i<m;i++)
```

```
                {
                        s[i].display();
        }
        System.out.println("\n *****TECHNICAL STAFF *****");
        System.out.println("SID\tNAME\tPHONE\t\tSALARY\tSKILLS");
                System.out.println("------");
                for(int i=0;i<n;i++)
                {
                         t[i].display();
                }




        System.out.println("\n *****CONTRACTOR STAFF *****");
        System.out.println("SID\tNAME\tPHONE\t\tSALARY\tPERIOD");
        System.out.println("------");
        for(int i=0;i<o;i++)
        {
                        c[i].display();
        }
    }
 }
```

OUTPUT:

```
******read teaching staff detail******
how many number of teaching staff:
2
enter teaching staff1details
enter the staffid
101
```

```
enter the name
Tanu
enter the phone
9449389269
enter the salary
25000
enter the domain
ISE
enter the publication
2
enter teaching staff2details
enter the staffid
102
enter the name
Ramya
enter the phone
9449489267
enter the salary
30000
enter the domain
CSE
enter the publication
2



******read technical staff detail******
how many number of technical staff:
2
```

```
enter technical staff1details
enter the staffid
401
enter the name
Bhagya
enter the phone
9876543478
enter the salary
15000
enter the number of skills
2
enter skills1
c
enter skills2
java
enter technical staff2details
enter the staffid
402
enter the name
Reshma
enter the phone
9876567899
enter the salary
20000

enter the number of skills
1
enter skills1
Java
******read contractor staff detail******
```

```
how many number of contractor staff:
2
enter contractor staff1details
enter the staffid
501
enter the name
Ram
enter the phone
9164575802
enter the salary
10000
how many years of contract
2
enter contractor staff2details
enter the staffid
502

enter the name
Manju
enter the phone
9876543457
enter the salary
15000
how many years of contract
3


            *****TEACHING STAFF *****
SID    NAME        PHONE            SALARY          DOMAIN      PUBLICATION
------
101    Tanu        9449389269   25000       ISE         2
102    Ramya       9449489267   30000       CSE         2
```

```
          *****TECHNICAL STAFF *****

SID    NAME          PHONE              SALARY             SKILLS

------

401    Bhagya              9876543478    15000       c,java,

402    Reshma        9876567899  20000          Java


       *****CONTRACTOR STAFF *****

SID    NAME          PHONE              SALARY             PERIOD

------

501    Ram           9164575802  10000          2

502    Manju         9876543457  15000          3
```

2b. 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 "/".

```
package javaapplication4;
import java.util.Scanner;
import java.util.StringTokenizer;
 class Customer
 {
	 String name;
	 String dob;
	 void read(String name, String dob)
	{
		this.name = name;
		this.dob = dob;
	}
	void display()
	{
		StringTokenizer st=new StringTokenizer(this.dob,"/");
		System.out.print(this.name+" ");
		while(st.hasMoreTokens())
		{
			System.out.print(","+st.nextToken());
		}
	}
}//End of class Customer

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

```
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Name :-");
                String name=sc.nextLine();
                System.out.println("Enter Date of birth(dd/mm/yyyy):-");
                String db=sc.next();
                Customer c=new Customer();
                c.read(name, db);
                c.display();
        }
}
```

OUTPUT 1:

```
Enter Name :-KALAM
Enter Date of birth:-19/01/2000
KALAM, 19 ,01, 2000
```

3a. 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.

```
package javaapplication5;
import java.util.Scanner;
class Compute
{
        private int a,b;
        public Compute(int m, int n)
        {
                a = m;
                b = n;
         }
        public void compute_a_by_b()
        {
                Try
                 {
                         if (b != 0)
                        {
                                System.out.println("Result a/b="+(float)1.0*a/b);
                         }
                         else throw new ArithmeticException("Denominator is 0! Division by zero
ERROR");
                 }
                catch (ArithmeticException e)
                {
                        System.out.println("Error !!!!: " + e);
                }
        }
 }
public class Lab3A
{
        public static void main(String[] args)
        {
                Scanner in = new Scanner(System.in);
                System.out.print("Enter a :-");
                int a = in.nextInt();
                System.out.print("Enter b:-");
                int b = in.nextInt();
                Compute compute = new Compute(a, b);
                compute.compute_a_by_b();
        }
}
```

OUTPUT 1:

```
Enter a :- 25
Enter b:- 2
Result a/b=12.5
```

3b. 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 and prints; third thread will print the value of cube of the number.

```
package javaapplication6;
import java.util.Random;
class Rand extends Thread
{
        public void run()
        {
                Random ra=new Random();
                int i=0;
                        while(i<10)
                        {
```

```
                        System.out.println(ra.nextInt(100)+"");
                        i++;
                        }
        }
}
class sn extends Thread
{
        int n;
        public sn(int n)
        {
                this.n=n;
        }
        public void run()
        {
                System.out.println("Square of number is:"+Math.pow(n,2));
        }
}
class cn extends Thread
{
        int n;
        public cn(int n)
        {
                this.n=n;
        }
        public void run()
        {
                System.out.println("Cube of number is:"+Math.pow(n,3));
        }
}
```

```
public class App
{
        public static void main(String[] args)
        {
                Rand r=new Rand();
                r.start();
                sn s1=new sn(3);
                s1.start();
                cn c1=new cn(2);
                c1.start();
        }

}
```

OUTPUT :1

```
49      69      68      73      72      59      94      84      62      73
Square of number is:9.0
Cube of number is:8.0
```

4. Sort a given set of n integer elements using the Quicksort 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

```
package javaapplication7;
import java.util.Random;
import java.util.Scanner;
public class quick
{
        static int max=2000;
        int partition (int[] a, int low,int high)
        {
                int p,i,j,temp;
                p=a[low];
                i=low+1;
                j=high;
                while(low<high)
                {
                        while(a[i]<=p&&i<high)
                 i++;
                        while(a[j]>p)
                        j--;
                        if(i<j)
                        {
                                temp=a[i];
                                a[i]=a[j];
                                a[j]=temp;
                        }
                 else
                        {
                                temp=a[low];
                                a[low]=a[j];
                                a[j]=temp; return j;
                        }
                }
                return j;
        }
        void sort(int[] a,int low,int high)
        {
                if(low<high)
                {
                                int s=partition(a,low,high);
                                sort(a,low,s-1);
                                sort(a,s+1,high);
                }
        }

        public static void main(String[] args)
        {
                int[] a; int i;
                System.out.println("Enter the array size");
                Scanner sc =new Scanner(System.in);
                int n=sc.nextInt();
                a= new int[max];
                Random generator=new Random();
                for( i=0;i<n;i++)
                a[i]=generator.nextInt(100);
                System.out.println("Array before sorting");
                for( i=0;i<n;i++)
                System.out.print(a[i]+" ");
                long startTime=System.nanoTime();
```

```
                quick m=new quick();
                m.sort(a,0,n-1);
                long stopTime=System.nanoTime();
                long elapseTime=(stopTime-startTime);
                System.out.println("\nTime  taken  to  sort  array  is: "+elapseTime+"
        nanoseconds");
                System.out.println("Sorted array is");
                for(i=0;i<n;i++)
                System.out.print(a[i]+" ");
        }

}
```

OUTPUT 1:

```
Enter the array size
10
Array before sorting
47 78 27 10 20 3 93 98 76 24
Time  taken  to  sort  array  is: 16768 nanoseconds
Sorted array is
3 10 20 24 27 47 76 78 93 98
```

OUTPUT 2:

```
Enter the array size
25
Array before sorting
21 8 97 52 3 78 46 80 71 81 33 87 99 42 73 30 90 56 8 34 21 97 28 73 66
Time  taken  to  sort  array  is: 25123 nanoseconds
Sorted array is
3 8 8 21 21 28 30 33 34 42 46 52 56 66 71 73 73 78 80 81 87 90 97 97 99
```

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.

```
package javaapplication8;
import java.util.Random;
import java.util.Scanner;
public class mergesort
{
        static int max=10000;
        void merge( int[] array,int low, int mid,int high)
        {
                int i=low;
                int j=mid+1;
                int k=low;
                int[]resarray;
                resarray=new int[max];
                while(i<=mid&&j<=high)
                {
                                if(array[i]<array[j])
                                {
                                        resarray[k]=array[i];
                                        i++;
                                        k++;
                                }
                                else
                                {
                                        resarray[k]=array[j];
                                        j++;
```

```
                    k++;
                }
        }
        while(i<=mid)
        resarray[k++]=array[i++];
        while(j<=high)
        resarray[k++]=array[j++];
        for(int m=low;m<=high;m++)
        array[m]=resarray[m];
}
```

```
void sort( int[] array,int low,int high)
{
        if(low<high)
        {


                int mid=(low+high)/2;
                        sort(array,low,mid);
                        sort(array,mid+1,high);
                        merge(array,low,mid,high);
        }
}
public static void main(String[] args)
{
        int[] array;
```

```
                    int i;
                    System.out.println("Enter the array size");
                    Scanner sc =new Scanner(System.in);
                    int n=sc.nextInt(); array= new int[max];
                    Random generator=new Random();
                    for( i=0;i<n;i++)
                    array[i]=generator.nextInt(20);
                    System.out.println("Array before sorting");
                    for( i=0;i<n;i++)
                    System.out.print(array[i]+" ");
                    long startTime=System.nanoTime();
                    mergesort m=new mergesort();
                    m.sort(array,0,n-1);
                    long stopTime=System.nanoTime();
                    long elapseTime=(stopTime-startTime);
                    System.out.println("\nTimetaken  to  sort  array  is:"+elapseTime+"nanoseconds");
                    System.out.println("Sorted array is");
                    for(i=0;i<n;i++)
                    System.out.print(array[i]+" ");
          }
}
```

OUTPUT 1:

```
Enter the array size
10
Array before sorting
6 15 14 13 16 14 6 16 6 10
Time taken  to  sort  array  is:7187796 nanoseconds
Sorted array is
6 6 6 10 13 14 14 15 16 16
```

OUTPUT 2:

```
Enter the array size
25
Array before sorting
44 94 9 73 31 73 67 28 91 71 57 66 89 26 25 80 72 77 4 53 59 73 66 6 72
Time taken  to  sort  array  is: 10120257nanoseconds
Sorted array is
4 6 9 25 26 28 31 44 53 57 59 66 66 67 71 72 72 73 73 73 77 80 89 91 94
```

6A. Implement in Java, the 0/1 Knapsack problem using Dynamic Programming method

```
package javaapplication9;
import java.util.Scanner;
public class knapsackdp
{
        public void solve(int[] wt, int[] val, int W, int N)
        {
                int i,j;
                int sol[][] = new int[N + 1][W + 1];
                for ( i = 0; i <= N; i++)
                {
                        for ( j = 0; j <= W; j++)
                        {
                                if(i==0||j==0)
                                sol[i][j]=0;
                                else if(wt[i]>j)
                                sol[i][j]=sol[i-1][j];
                                else
                                sol[i][j]=Math.max((sol[i-1][j]), (sol[i - 1][j - wt[i]] + val[i]));
                        }
                }
                System.out.println("The optimal solution is"+sol[N][W]);
                int[] selected = new int[N + 1];
                for(i=0;i<N+1;i++)
                selected[i]=0;
                i=N;
                j=W;
                while (i>0&&j>0)
                {
                        if (sol[i][j] !=sol[i-1][j])
                        {
                                selected[i] = 1; j = j - wt[i];
                        }
                        i--;
                }
                System.out.println("\nItems selected : ");
                for ( i = 1; i < N + 1; i++)
                if (selected[i] == 1)
                System.out.print(i +" ");
                System.out.println();
        }
        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);
                knapsackdp ks = new knapsackdp();


                System.out.println("Enter number of elements ");
                int n = scan.nextInt();
                int[] wt = new int[n + 1];
                int[] val = new int[n + 1];
                System.out.println("\nEnter weight for "+ n +" elements");
                for (int i = 1; i <= n; i++)
                wt[i] = scan.nextInt();
                System.out.println("\nEnter value for "+ n +" elements");
                for(int i = 1; i <= n; i++)
                val[i] = scan.nextInt();
                System.out.println("\nEnter knapsack weight ");
                int W = scan.nextInt();
                ks.solve(wt, val, W, n);
        }
}
```

OUTPUT :1

```
Enter number of elements
4
Enter weight for 4 elements
2       1       3       2
Enter value for 4 elements
12
10
20
15
Enter knapsack weight
5
The optimal solution is 37
Items selected :
1       2       4
```

6B. Implement in Java, the 0/1 Knapsack problem using Greedy method

```
package javaapplication10;
import java.util.Scanner;
public class knapsackgreedy
{
        public static void main(String[] args)
        {
                int i,j=0,max_qty,m,n;
                float sum=0,max;
                Scanner sc = new Scanner(System.in);
                int array[][]=new int[2][20];
                System.out.println("Enter no of items");
                n=sc.nextInt();
                System.out.println("Enter the weights of each items");
                for(i=0;i<n;i++)
                array[0][i]=sc.nextInt();
                System.out.println("Enter the values of each items");
                for(i=0;i<n;i++)
                array[1][i]=sc.nextInt();
                System.out.println("Enter maximum volume of knapsack :");
                max_qty=sc.nextInt();
                m=max_qty; while(m>=0)
                {
                        max=0; for(i=0;i<n;i++)
                        {
                                if(((float)array[1][i])/((float)array[0][i])>max)
                                {
                                        max=((float)array[1][i])/((float)array[0][i]);
                                        j=i;
                                }
                        }
                        if(array[0][j]>m)
                        {
                        System.out.println("Quantity of item number: "+     (j+1) + " added is " +m);
                        sum+=m*max; m=-1;
                        }
                        else
                        {
                        System.out.println("Quantity of item number: " + (j+1) + " added is " +
array[0][j]);
                        m-=array[0][j]; sum+=(float)array[1][j];
                        array[1][j]=0;
                }
        }
        System.out.println("The total profit is " + sum); sc.close();
    }
}
```

OUTPUT :1

```
Enter no of items
4
Enter the weights of each items
2
1
3
2
Enter the values of each items
12
10
20
15
Enter maximum volume of knapsack :
```

```
5
Quantity of item number: 2 added is 1
Quantity of item number: 4 added is 2
Quantity of item number: 3 added is 2

The total profit is 38.333332
```

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.

```
package javaapplication10;
import java.util.Scanner;
public class Dijkstra
{
        int d[]=new int[10]; int p[]=new int[10];
        int visited[]=new int[10];
        public void dijk(int[][]a, int s, int n)
        {
                int u=-1,v,i,j,min;
                for(v=0;v<n;v++)
                {
                        d[v]=99;
                        p[v]=-1;
                }
                d[s]=0;
                for(i=0;i<n;i++)
                {
                        min=99;
                        for(j=0;j<n;j++)
                        {
                                if(d[j]<min&& visited[j]==0)
                                {
                                        min=d[j];
                                        u=j;
                                }
                        }
                        visited[u]=1;
                        for(v=0;v<n;v++)
                        {
                                if((d[u]+a[u][v]<d[v])&&(u!=v)&&visited[v]==0)
```

```
                    {
                            d[v]=d[u]+a[u][v]; p[v]=u;
                    }
                }
            }
        }
        void path(int v,int s)
        {
            if(p[v]!=-1) path(p[v],s);
            if(v!=s)
            System.out.print("->"+v+" ");
        }



        void display(int s,int n)
        {
            int i;
            for(i=0;i<n;i++)
            {
                if(i!=s)
                {
                    System.out.print(s+" ");
                    path(i,s);
                }
                if(i!=s)
                System.out.print("="+d[i]+" ");
                System.out.println();
            }
        }
        public static void main(String[] args)
        {
            int a[][]=new int[10][10];
            int i,j,n,s;
            System.out.println("enter the number of vertices");
            Scanner sc = new      Scanner(System.in);
            n=sc.nextInt();
            System.out.println("enter the weighted matrix");
            for(i=0;i<n;i++)
            for(j=0;j<n;j++)
            a[i][j]=sc.nextInt();
            System.out.println("enter the source vertex"); s=sc.nextInt();
            Dijkstra tr=new Dijkstra();
            tr.dijk(a,s,n);
            System.out.println("the shortest path between source"+s+"to remaining vertices are");
            tr.display(s,n);
            sc.close();
        }
   }
```

OUTPUT:

```
enter the number of vertices
```

```
5
enter the weighted matrix
0 3 99 7 99
3 0 4 2 99
99 4 0 5 6
5 2 5 0 4
99 99 6 4 0
enter the source vertex
0
the shortest path between source0to remaining vertices are

0 ->1 =3
0 ->1 ->2 =7
0 ->1 ->3 =5
0 ->1 ->3 ->4 =9
```

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

```
package javaapplication12;
import java.util.Scanner;
public class kruskal
{
        int parent[]=new int[10];
        int find(int m)
        {
                int p=m;
                while(parent[p]!=0)
                p=parent[p];
                return p;
        }
        void union(int i,int j)
        {
                if(i<j) parent[i]=j;
                else
                parent[j]=i;
    }
    void krkl(int[][]a, int n)
    {
                int u=0,v=0,min,k=0,i,j,sum=0;
                while(k<n-1)
                {
                        min=99;
                        for(i=1;i<=n;i++)
                        for(j=1;j<=n;j++)
                        if(a[i][j]<min&&i!=j)
                        {
                                min=a[i][j];
                                u=i;
```

```
                    v=j;
                }
                i=find(u);
                j=find(v);
                if(i!=j)
                {
                    union(i,j);
                    System.out.println("("+u+","+v+")"+"="+a[u][v]);
                    sum=sum+a[u][v];
                    k++;
                }
                a[u][v]=a[v][u]=99;
            }


            System.out.println("The cost of minimum spanning tree = "+sum);
        }
        public static void main(String[] args)
        {
            int a[][]=new int[10][10];
            int i,j;
            System.out.println("Enter the number of vertices of the graph");
            Scanner sc=new Scanner(System.in);
            int n; n=sc.nextInt();
            System.out.println("Enter the wieghted matrix");
            for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
            a[i][j]=sc.nextInt();
            kruskal k=new kruskal();
            k.krkl(a,n);
            sc.close();
        }
```

```
 }
```

OUTPUT:

```
Enter the number of vertices of the graph
6
Enter the wieghted matrix
0 3 99 99 6 5
3 0 1 99 99 4
99 1 0 6 99 4
99 99 6 0 8 5
6 99 99 8 0 2
5  4  4 5 2 0
(2,3)=1
(5,6)=2
(1,2)=3
(2,6)=4
(4,6)=5
The cost of minimum spanning tree = 15
```

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

```
package javaapplication13;
java.util.Scanner;
public class prims
{
        public static void main(String[] args)
        {
                int w[][]=new int[10][10];
                int n,i,j,s,k=0; int min;
                int sum=0; int u=0,v=0; int flag=0;
                int sol[]=new int[10];
                System.out.println("Enter the number of vertices");
                Scanner sc=new Scanner(System.in);
                n=sc.nextInt();
                for(i=1;i<=n;i++)
                sol[i]=0;
                System.out.println("Enter the weighted graph");
                for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                 w[i][j]=sc.nextInt();
                System.out.println("Enter the source vertex");
                s=sc.nextInt();
                sol[s]=1;
                k=1;
                while (k<=n-1)
                {
                        min=99;
                        for(i=1;i<=n;i++)
                        for(j=1;j<=n;j++)
                         if(sol[i]==1&&sol[j]==0)
                         if(i!=j&&min>w[i][j])
                        {
```

```
                    min=w[i][j];
                    u=i;
                    v=j;
                }
                sol[v]=1;
                sum=sum+min;
                k++;
                System.out.println(u+"->"+v+"="+min);
            }
            for(i=1;i<=n;i++)
            if(sol[i]==0)
            flag=1;
            if(flag==1)


            System.out.println("No spanning tree");
            else
            System.out.println("The cost of minimum spanning tree is"+sum);
            sc.close();
        }
}
```

OUTPUT:

```
Enter the number of vertices 6
Enter the weighted graph
0 3 99 99 6 5
3 0 1 99 99 4
99 1 0 6 99 4
99 99 6 0 8 5
6 99 99 8 0 2
5 4  4  5 2 0
Enter the source vertex 1
```

```
1->2=3

2->3=1

2->6=4

6->5=2

6->4=5
```

10A. Write Java programs to implement All-Pairs Shortest Paths problem using Floyd's algorithm.

```
package javaapplication14;
import java.util.Scanner;
public class Flyd
{
        void flyd(int[][] w,int n)
        {
                int i,j,k; for(k=1;k<=n;k++)
                for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                w[i][j]=Math.min(w[i][j], w[i][k]+w[k][j]);
        }
        public static void main(String[] args)
        {
                int a[][]=new int[10][10];
                int n,i,j;
                System.out.println("enter the number of vertices");
                Scanner sc=new Scanner(System.in);
                n=sc.nextInt();
                System.out.println("Enter the weighted matrix");
                for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                a[i][j]=sc.nextInt();
                Flyd f=new Flyd();
                f.flyd(a, n);
                System.out.println("The shortest path matrix is");
                for(i=1;i<=n;i++)
                {
                        for(j=1;j<=n;j++)
                        {
```

```
                                        System.out.print(a[i][j]+" ");
                                }
                                System.out.println();
                        }
                        sc.close();
                }

        }
```

OUTPUT:

```
enter the number of vertices
4
Enter the weighted matrix
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0
The shortest path matrix is
0 10 3 4
2 0 5 6
7 7 0 1
6 16 9 0
```

10B. Write Java programs to implement Travelling Sales Person problem using Dynamic programming.

```
package javaapplication15;
import java.util.Scanner;
public class lab10b
{
        public static void main(String[] args)
        {
                 int c[][]=new int[10][10], tour[]=new int[10];
                Scanner in = new Scanner(System.in);
                int i, j,cost;
                System.out.println("**** TSP  DYNAMIC  PROGRAMMING *******");
                System.out.println("Enter the number of cities: ");
                int n = in.nextInt();
                if(n==1)
                {
                         System.out.println("Path is not possible");
                        System.exit(0);
                 }
                System.out.println("Enter the cost matrix");
                for(i=1;i<=n;i++)
                 for(j=1;j<=n;j++)
                        c[i][j] = in.nextInt();
                System.out.println("The entered cost matrix is");
                 for(i=1;i<=n;i++)
                 {
                        for(j=1;j<=n;j++)
                         {
                                        System.out.print(c[i][j]+"\t");
                         }
                        System.out.println();
```

```
            }
             for(i=1;i<=n;i++)
            tour[i]=i;
            cost = tspdp(c, tour, 1, n);
            System.out.println("The accurate path is");
            for(i=1;i<=n;i++)
            System.out.print(tour[i]+"->");
            System.out.println("1");
            System.out.println("The accurate mincost is "+cost);
            System.out.println("******* ************* ***************");
    }




static int tspdp(int c[][], int tour[], int start, int n)
{
 int mintour[]=new int[10], temp[]=new int[10], mincost=999, ccost, i, j, k;
             if(start == n-1)
             {
                     return (c[tour[n-1]][tour[n]] + c[tour[n]][1]);
            }
            for(i=start+1; i<=n; i++)
            {
                    for(j=1; j<=n; j++)
                    temp[j] = tour[j];
                    temp[start+1] = tour[i];
                    temp[i] = tour[start+1];
                    if((c[tour[start]][tour[i](ccost=tspdp(c,temp,start+1,n)))<mincost)
                    {
                            mincost = c[tour[start]][tour[i]] + ccost;
                    for(k=1; k<=n; k++)
                    mintour[k] = temp[k];
               }
          }
          for(i=1; i<=n; i++)
          tour[i] = mintour[i];
          return mincost;
     }
}
```

OUTPUT:

```
**** TSP  DYNAMIC  PROGRAMMING *******
Enter the number of cities:
5
Enter the cost matrix
0       8       99      99      5
12      0       99      16      99
99      7       0       8       20
1       6       19      0       3
9       99      12      18      0
The entered cost matrix is
0       8       99      99      5
12      0       99      16      99
99      7       0       8       20
1       6       19      0       3
9       99      12      18      0
The accurate path is
1->5->3->2->4->1
The accurate mincost is 41
******* ************* ***************
```

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.

```
package javaapplication16;
import java.util.Scanner;
import static java.lang.Math.pow;
public class subset
{
        void subset(int num,int n, int x[])
        {
                int i;
                for(i=1;i<=n;i++)
                x[i]=0;
                for(i=n;num!=0;i--)
                {
                                x[i]=num%2;
                                num=num/2;
                }
        }
        public static void main(String[] args)
        {
                 int x[]=new int[10];
                 int n,d,sum,present=0;
                 int a[]=new int[10];
                 int j;
                System.out.println("enter the number of elements of set"); Scanner sc=new
Scanner(System.in);
                n=sc.nextInt();
                System.out.println("enter the elements of set");
                 for(int i=1;i<=n;i++)
                a[i]=sc.nextInt();
                System.out.println("enter the positive integer sum");
                d=sc.nextInt();
                 if(d>0)
                {
                                for(int i=1;i<=Math.pow(2,n)-1;i++)
                                {
                                        subset s=new subset();
                                        s.subset(i,n,x);
                                        sum=0;
                                        for(j=1;j<=n;j++)
                                        if(x[j]==1)
                                        sum=sum+a[j];




                                if(d==sum)
                                        {
                                                System.out.print("Subset={");
                                                present=1;
                                                for(j=1;j<=n;j++)
                                                if(x[j]==1)
                                                System.out.print(a[j]+",");
                                                System.out.print("}="+d);
                                                System.out.println();
                                        }
                                }
                }
                 if(present==0)
                System.out.println("Solution does not exists");
        }
```

```
}
```

OUTPUT 1:

```
enter the number of elements of set
5
enter the elements of set
10
20
30
40
50
enter the positive integer sum
100
Subset={20,30,50,}=100
Subset={10,40,50,}=100
Subset={10,20,30,40,}=100
```

OUTPUT 2:

```
enter the number of elements of set
5
enter the elements of set
1
2
3
4
5
enter the positive integer sum
25
Solution does not exists
```

12. Design and implement the presence of Hamiltonian Cycle in an undirected Graph G of *n* vertices.

```
package javaapplication44;
import java.util.*;
class Hamiltoniancycle1
{
        private int adj[][],x[],n;
        public Hamiltoniancycle1()
        {
                Scanner src = new Scanner(System.in);
                System.out.println("Enter the number of nodes");
                n=src.nextInt();
                x=new int[n];
                x[0]=0;
                for (int i=1;i<n; i++)
                x[i]=-1;
                adj=new int[n][n];
                System.out.println("Enter the adjacency matrix");
                for (int i=0;i<n; i++)
                for (int j=0; j<n; j++)
                adj[i][j]=src.nextInt();
        }
        public void nextValue (int k)
        {
                int i=0; while(true)
                {
                        x[k]=x[k]+1;
                        if (x[k]==n)
                        x[k]=-1;
                        if (x[k]==-1)
                        return;
```

```
                    if (adj[x[k-1]][ x[k]]==1)
                    for (i=0; i<k; i++)
                    if(x[i]==x[k])
                    break;
                    if (i==k)
                    if (k<n-1 || k==n-1 && adj[x[n-1]][0]==1)
                    return;
                }
        }
        public void getHCycle(int k)
        {
                while(true)
                {
                    nextValue(k);
                    if (x[k]==-1)
                    return;
                    if (k==n-1)
                    {
                        System.out.println("\nSolution : ");
                        for (int i=0; i<n; i++)
                        System.out.print((x[i]+1)+" ");
                        System.out.println(1);
                    }
                    else getHCycle(k+1);
                }

        }

}
class Hamiltoniancycle
{
        public static void main(String args[])
        {
                Hamiltoniancycle1 obj=new Hamiltoniancycle1();
                obj.getHCycle(1);
        }
}
```

<u>OUTPUT :</u>

```
Enter the number of nodes
5
Enter the adjacency matrix
0 1 1 1 0
1 0 1 0 1
1 1 0 1 0
1 0 1 0 1
0 1 0 1 0
Solution :
1 2 5 4 3 1

Solution :
1 3 2 5 4 1

Solution :
1 3 4 5 2 1

Solution :
1 4 5 2 3 1
```