Program 1

1a. 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

```
class student
{
	String USN, Name, Branch, Phone;
	student(String U, String N, String B, String P)
	{
		this.USN = U;
		this.Name = N;
		this.Branch = B;
		this.Phone = P;
	}

	void display()
	{
		System.out.println("\nStudent details");
		System.out.println("USN = "+this.USN);
		System.out.println("Name = "+this.Name);
		System.out.println("Branch = "+this.Branch);
		System.out.println("Phone = "+this.Phone);
	}
}

class Prg1a {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		student s1 = new student("1RN17IS166","Nagraj","ISE","9933");
		student s2 = new student("1RN17IS196","Sindhraj","ISE","7711");
		s1.display();
		s2.display();
	}
}
```

OUTPUT:

```
Student details
USN = 1RN17IS166
Name = Nagraj
Branch = ISE
Phone = 9933

Student details
USN = 1RN17IS196
Name = Sindhraj
Branch = ISE
Phone = 7711
```

Alternative program

1a. 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

```
import java.util.Scanner;
class  Stud{
        String USN, Name, Branch, Phone;
        void read()
        {
          Scanner sc = new Scanner(System.in);
          System.out.println("Enter the USN: ");
          USN  =sc.next();
         System.out.println("Enter the Name: ");
         Name  =sc.next();
         System.out.println("Enter the Branch: ");
          Branch = sc.next();
        System.out.println("Enter the Phone No.: ");
        Phone  =sc.next();
}
        void display()
        {
                System.out.println("\nStudent details");
                System.out.println("USN = "+this.USN);
                System.out.println("Name = "+this.Name);
                System.out.println("Branch = "+this.Branch);
                System.out.println("Phone = "+this.Phone);
        }
}
class Prg1a_inp {
        public static void main(String[] args) {

                Scanner sc = new Scanner(System.in);
                System.out.println("Enter the number of students:");
                int nos = sc.nextInt();
                Stud [] s = new Stud[nos];
                for(int i = 0;i<nos;i++)
                {
                        System.out.println("Enter the details for student "+(i+1));
                        s[i] = new Stud();
                        s[i].read();
                }
                System.out.println("The student details are:  ");
                for(int i = 0;i<nos;i++)
                {
                        System.out.println("Enter the details for student "+(i+1));
                        s[i].display();
                }

        sc.close();
```

```
        }
}
```

OUTPUT:

```
Enter the number of students:
2
Enter the details for student 1
Enter the USN:
1RN14IS001
Enter the Name:
ABHI
Enter the Branch:
ISE
Enter the Phone No.:
9877568988
Enter the details for student 2
Enter the USN:
1RN15IS001
Enter the Name:
AKHI
Enter the Branch:
ISE
Enter the Phone No.:
1234567890
The student details are:
Enter the details for student 1

Student details
USN = 1RN14IS001
Name = ABHI
Branch = ISE
Phone = 9877568988
Enter the details for student 2

Student details
USN = 1RN15IS001
Name = AKHI
Branch = ISE
Phone = 1234567890
```

Stack Operations

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

```
import java.util.Scanner;
class Stack{
        Scanner sc = new Scanner(System.in);
        int s[], top, size;
        Stack(int n){
                size = n;
                s = new int[size];
```

```
		top = -1;
	}
	void push(){
		if(top == size-1)
			System.out.println("stack is full");
		else
		{
			System.out.println("Enter element to push:");
			int element = sc.nextInt();
			s[++top] = element;
		}
	}
	int pop(){
		if(top == -1) {
			System.out.println("stack is empty.");
			return -1;
		}
		else
			return (s[top--]);
	}

	void display(){
		if(top == -1)
			System.out.println("stack is empty");
		else
		{
			System.out.println("stack elements are :");
			for(int i=top;i>=0;i--)
				System.out.println(s[i]);
		}
	}
}
class prg1b {
	public static void main(String[] args) {
		System.out.println("Enter stack size:");
		Scanner sc = new Scanner(System.in);
		int n = sc.nextInt();
		int choice;
		Stack stkOb = new Stack(n);
		do{
			System.out.println("Stack operations:");
			System.out.println("1. Push");
			System.out.println("2. Pop");
			System.out.println("3. Display");
			System.out.println("4. Exit");
			System.out.println("Enter your choice.");
			choice = sc.nextInt();
			switch (choice)
			{
				case 1:stkOb.push();break;
				case 2:int popped = stkOb.pop();
```

```
                                        if(popped != -1 )
                                        System.out.println("Popped element is :"+ popped);
                                                break;
                                case 3:stkOb.display(); break;
                                case 4: System.out.println("Thank You");break;
                        }
                }while(choice != 4);
        }
}
```

OUTPUT:

```
Enter stack size:
3
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
1
Enter element to push:
10
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
1
Enter element to push:
20
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
1
Enter element to push:
30
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
1
stack is full
Stack operations:
1. Push
2. Pop
```

```
3. Display
4. Exit
Enter your choice.
3
stack elements are :
30
20
10
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
2
Popped element is :30
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
2
Popped element is :20
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
2
Popped element is :10
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
2
stack is empty.
Stack operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice.
3
stack is empty
Stack operations:
1. Push
2. Pop
```

3. Display
4. Exit
Enter your choice.
4
Thank You

Program 2

2a. Design a super class called Staff with details as Staff Id, 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.

```
import java.util.Scanner;

class staff {
	String name, phone;
	int sid, sal;
	Scanner scan = new Scanner(System.in);

	void read() {
		System.out.println("Staff Id:");
		sid = scan.nextInt();
		System.out.println("Name:");
		name = scan.next();
		System.out.println("Phone:");
		phone = scan.next();
		System.out.println("salary:");
		sal = scan.nextInt();
		scan.nextLine(); //additional dummy scan to flush the buffer contents
		// if this scan.nextLine(), is not used input of string after the
		// integer in the console, the string input is just skipped
	}

	void display() {
		System.out.println("Staff Id:" + sid);
		System.out.println("Name:" + name);
		System.out.println("Phone:" + phone);
		System.out.println("salary:" + sal);
	}
}

class teaching extends staff {
	String domain;
	int publication;

	void read() {
		super.read();
		System.out.print("Enter the Domain Expertise:");
		domain = scan.nextLine();
		System.out.print("Enter the No. of publications:");
		publication = scan.nextInt();
	}
```

```
	void display() {
		super.display();
		System.out.println("Domain Expert is :" + domain);
		System.out.println("No. of Publications :" + publication);
	}
}

class technical extends staff {
	String[] skills;

	void read() {
		super.read();
		System.out.print("Skills (separated by commas):");
		skills = scan.nextLine().split(",");
	}

	void display() {
		super.display();
		System.out.print("Skills:");
		for (int i = 0; i<skills.length; i++) {
			System.out.print(skills[i] + " ");
		}
		System.out.println();
	}
}

class contract extends staff {
	int period;

	void read() {
		super.read();
		System.out.print("Enter the Contract period: ");
		period = scan.nextInt();
	}

	void display() {
		super.display();
		System.out.print("Contract Period: " + period);
	}
}

class prg2a {
	public static void main(String[] args) {
		teaching teach[] = new teaching[3];
		technical tech[] = new technical[3];
		contract cont[] = new contract[3];
		// read 3 staff objects of all three categories.
		System.out.println("Enter the staff details");
		for (int i = 0; i< 3; i++) {
			System.out.println("\n Enter the details of teaching staff # " + (i + 1));
			teach[i] = new teaching();
```

```
                teach[i].read();
            }
            for (int i = 0; i< 3; i++) {
                System.out.println("\n Enter the details of technical staff # " + (i + 1));
                tech[i] = new technical();
                tech[i].read();
            }
            for (int i = 0; i< 3; i++) {
                System.out.println("\n Enter the details of contract staff # " + (i + 1));
                cont[i] = new contract();
                cont[i].read();
            }

            // display 3 staff objects of all three categories.
            System.out.println("\n The details of all three categories of staffs are\n");
            for (int i = 0; i< 3; i++) {
                System.out.println("\n The details of teaching staff # " + (i + 1));
                teach[i].display();
            }
            for (int i = 0; i< 3; i++) {
                System.out.println("\n The details of teaching staff # " + (i + 1));
                tech[i].display();
            }
            for (int i = 0; i< 3; i++) {
                System.out.println("\n The details of teaching staff # " + (i + 1));
                cont[i].display();
            }
        }
}
```

```
Enter the staff details

 Enter the details of teaching staff # 1
Staff Id:
1
Name:
kumar
Phone:
123456789
salary:
12000
Enter the Domain Expertise:Computer Networks
Enter the No. of publications:12

 Enter the details of teaching staff # 2
Staff Id:
1
Name:
Madhu
Phone:
9876543210
```

salary:
25000
Enter the Domain Expertise:Image Processing
Enter the No. of publications:10

Enter the details of teaching staff # 3
Staff Id:
3
Name:
Ashwini
Phone:
1234567789
salary:
20000
Enter the Domain Expertise:Sensor Networks
Enter the No. of publications:3

Enter the details of technical staff # 1
Staff Id:
1
Name:
usha
Phone:
123456788
salary:
20000
Skills (separated by commas):C,C++

Enter the details of technical staff # 2
Staff Id:
2
Name:
Uma
Phone:
4343434343
salary:
10000
Skills (separated by commas):java,python

Enter the details of technical staff # 3
Staff Id:
3
Name:
Manu
Phone:
1212121212
salary:
30000
Skills (separated by commas):Python, R

Enter the details of contract staff # 1

Staff Id:
1
Name:
Kushi
Phone:
121345666
salary:
5000
Enter the Contract period: 2

Enter the details of contract staff # 2
Staff Id:
2
Name:
Shree
Phone:
56565674
salary:
8000
Enter the Contract period: 5

Enter the details of contract staff # 3
Staff Id:
3
Name:
kavya
Phone:
123213422
salary:
3000
Enter the Contract period: 1

The details of all three categories of staffs are
The details of teaching staff # 1
Staff Id:1
Name:kumar
Phone:123456789
salary:12000
Domain Expert is :Computer Networks
No. of Publications :12

The details of teaching staff # 2
Staff Id:1
Name:Madhu
Phone:9876543210
salary:25000
Domain Expert is :Image Processing
No. of Publications :10

The details of teaching staff # 3
Staff Id:3

Name:Ashwini
Phone:1234567789
salary:20000
Domain Expert is :Sensor Networks
No. of Publications :3

The details of teaching staff # 1
Staff Id:1
Name:usha
Phone:123456788
salary:20000
Skills:C C++

The details of teaching staff # 2
Staff Id:2
Name:Uma
Phone:4343434343
salary:10000
Skills:java python

The details of teaching staff # 3
Staff Id:3
Name:Manu
Phone:1212121212
salary:30000
Skills:Python R

The details of teaching staff # 1
Staff Id:1
Name:Kushi
Phone:121345666
salary:5000
Contract Period: 2
The details of teaching staff # 2
Staff Id:2
Name:Shree
Phone:56565674
salary:8000
Contract Period: 5
The details of teaching staff # 3
Staff Id:3
Name:kavya
Phone:123213422
salary:3000
Contract Period: 1

2.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 "/".

```
import java.util.Scanner;
import java.util.StringTokenizer;

class StrToken2b
{
	private String name, date_of_birth;
	public void input() {
		Scanner s = new Scanner(System.in);
		System.out.println("enter the name");
		name = s.nextLine();
		System.out.println("enter the date of birth in format dd/mm/yyyy");
		date_of_birth = s.nextLine();
		s.close();
	}
	public void display()
	{
		System.out.print("Name and ");
		StringTokenizer st = new StringTokenizer(date_of_birth, "/");
		System.out.print("Date of birth is:"+ " < " + name + ", ");
		int n = st.countTokens();
		while (st.hasMoreTokens())
		{
			if(n>1)
				System.out.print(st.nextToken("/") + ", ");
			else
				System.out.println(st.nextToken("/") + " >");
			n--;
		}
	}
	public static void main(String[] args) {
		StrToken2b st = new StrToken2b();
		st.input();
		st.display();
	}
}
```

OUTPUT:

```
enter the name
Suma
enter the date of birth in format dd/mm/yyyy
30/05/1976
Name and Date of birth is: < Suma, 30, 05, 1976 >
```

Program 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.

```
import java.util.Scanner;
class prg3a {
	public static void main(String[] args)
	{
		int a,b;
		float res;
		try
		{
			Scanner inn=new Scanner(System.in);
			System.out.println("Input Dividend (a) \n");
			a=inn.nextInt();
			System.out.println("Input Divisor (b) \n");
			b=inn.nextInt();
			res=a/b;
			System.out.println("Quotient is="+res);
			inn.close();
		}
		catch(ArithmeticException e)
		{
			System.out.println("Divide by zero error "+e);
		}
	}
}
```

Output 1:

```
Input Dividend (a): 21
Input Divisor (b): 3
Quotient is = 7.0
```

Output 2:

```
Input Dividend (a): 5
Input Divisor (b): 0
Divide by zero error java.lang.ArithmeticException: / by zero
```

Program3b

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.

```
import java.util.Random;
//using Thread Class
class prg3b{
	static int randomInteger;
	public static void main(String[] args){
		System.out.println("For 10 Random numbers");
		MyThread1 thread1 = new MyThread1();
		thread1.start();//start thread1
```

```
	}
}
//Thread1
class MyThread1 extends Thread{
	public void run() {
		int i = 0;
		try {
			//generating 10 random numbers
			while(i<10) {
				Random random = new Random();
				prg3b.randomInteger = random.nextInt(10);
				System.out.println((i+1)+" Random integer is "+prg3b.randomInteger);
				new MyThread2().start();//start thread2
				new MyThread3().start();//start thread3
				Thread.sleep(1000*1);//delay for display of output
				System.out.println("\n");
				i++;
			}
		}
		catch (InterruptedException exception) {
			exception.printStackTrace();
		}
	}
}
//Thread2
class MyThread2 extends Thread {
public void run() {
System.out.println("Square of "+prg3b.randomInteger + " is " +prg3b.randomInteger * prg3b.randomInteger );
	}
}
//Thread3
class MyThread3 extends Thread {
public void run() {
System.out.println("Cube of "+prg3b.randomInteger + " is " + prg3b.randomInteger * prg3b.randomInteger *
			prg3b.randomInteger );
	}
}
```

```
For 10 Random numbers
1 Random integer is 5
Cube of 5 is 125
Square of 5 is 25

2 Random integer is 7
Cube of 7 is 343
Square of 7 is 49

3 Random integer is 1
Square of 1 is 1
Cube of 1 is 1
4 Random integer is 9
```

Square of 9 is 81
Cube of 9 is 729

5 Random integer is 3
Cube of 3 is 27
Square of 3 is 9

6 Random integer is 3
Square of 3 is 9
Cube of 3 is 27

7 Random integer is 9
Square of 9 is 81
Cube of 9 is 729

8 Random integer is 7
Square of 7 is 49
Cube of 7 is 343

9 Random integer is 6
Cube of 6 is 216
Square of 6 is 36

10 Random integer is 2
Square of 2 is 4
Cube of 2 is 8

Program 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 non 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.

```
import java.util.*;
import java.io.*;
class QSort
{
	int array[];
	int length;
	int cnt;
	int size;
	String flag;
	void process() throws IOException
	{
		Scanner scanner = new Scanner(System.in);
		size = scanner.nextInt();
		flag = "best"; // flag is set to true for ascending order sorting
		cnt = 0;
		int[] inputArr = new int[size];
```

```
		getRandomNumbers(inputArr);
		scanner.close();
		int tempArr[] = new int[size];
		for (int i =0; i<inputArr.length ;i++)
			tempArr[i] = inputArr[i];
		// Begin to sort the randomly generated numbers for best case complexity
		sort(inputArr);
		System.out.println();
		System.out.println("The time complexity for best case is " + cnt);
		PrintWriter outA = new PrintWriter(new File("q_asort.txt"));
		for (int i :inputArr)
			outA.print(i + " ");
		outA.close();
		cnt = 0;
		flag = "worst";// Begin to sort the sorted numbers for worst case complexity
		sort(inputArr);
		System.out.println();
		System.out.println("The time complexity for worst case is " + cnt);
		cnt = 0;
		flag = "average"; // flag is set to average case
		sort(tempArr);// sort the original array
		System.out.println();
		PrintWriter outb = new PrintWriter(new File("q_bsort.txt"));
		for (int i : inputArr) {
			outb.print(i);
			outb.print(" ");
		}
		outb.close();
		System.out.println("The time complexity for average case is " + cnt);
}
void getRandomNumbers(int[] inputArr) throws IOException
{
		Random rand = new Random();
		int number, count = 0;
		PrintWriter out = new PrintWriter(new File("Qrandom.txt"));
		while (count<size)
		{
			number = rand.nextInt(size) + 1;
			out.print(number + " ");
			inputArr[count] = number;
			count++;
		}
		out.close();
		System.out.println("The total numbers generated: " + count);
}
void sort(int[] inputArr)
{
		int pivot = 0;
		//long count = 0;
		if (inputArr == null || inputArr.length == 0) {
			return;
```

```
        }
        this.array = inputArr;
        length = inputArr.length;
        if (flag == "best" || flag == "average")
        {
                pivot = array[(length) / 2];
                quickSort(0, length - 1, pivot);
        }
        else if (flag == "worst")
        {
                pivot = array[0];
                quickSort(0, length - 1, pivot);
        }
}
void quickSort(int lowerIndex, int higherIndex, int pivot)
{
        int left = lowerIndex;
        int right = higherIndex;
        // calculate pivot number
        switch (flag) {
                case "best":
                case "average":
                pivot = array[lowerIndex + (higherIndex - lowerIndex) / 2];
                break;
                case "worst":
                pivot = array[lowerIndex];
                break;
        }
        while (left<= right) { // sort in ascending order
                while (array[left] <pivot) {
                        left++;
                        cnt++;
                }
                while (array[right] >pivot) {
                        right--;
                        cnt++;
                }
                if (left<= right) {
                        exchangeNumbers(left, right);
                        // move index to next position on both sides
                        left++;
                        right--;
                }
        }
        // call quickSort() method recursively
        if (lowerIndex<right) {
                quickSort(lowerIndex, right, pivot);
        }
        if (left<higherIndex) {
                quickSort(left, higherIndex, pivot);
        }
```

```
        }
        void exchangeNumbers(int l, int r) {
                int temp = array[l];
                array[l] = array[r];
                array[r] = temp;
        }
}
class QuickSort1 {
        public static void main(String args[]) throws IOException {
                System.out.print("Enter the number of elements to sorted: ( >5000):");
                QSort qs = new QSort();
                qs.process();
        }
}
```

Output 1:

Enter the number of elements to sorted: (>5000):10000
The total numbers generated: 10000

The time complexity for best case is 102321

The time complexity for worst case is 45893027

The time complexity for average case is 102321

Output 2:

Enter the number of elements to sorted: (>5000):15000
The total numbers generated: 15000

The time complexity for best case is 148081

The time complexity for worst case is 102731608

The time complexity for average case is 148081

Program 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 non 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.

```
import java.util.Scanner;
import java.util.Random;
import java.io.*;

class MergeSort {
```

```
int[] inputArr;
int[] tempMergArr;
int cnt;
int size;

void process() throws IOException {
	Scanner scanner = new Scanner(System.in);
	System.out.print("Enter the number of elements to sorted:(>5000):");
	size = scanner.nextInt();
	inputArr = new int[size];
	tempMergArr = new int[size];
	getRandomNumbers(inputArr);
	cnt = 0; // the code computes time complexity for all cases ...
	doMergeSort(0, inputArr.length - 1);

	PrintWriter outA = new PrintWriter(new File("mrg_asort.txt"));
	for (int i :inputArr)
		outA.print(i + " ");
	outA.close();
	System.out.println();
	System.out.println("The time complexity is ... " + cnt);
	scanner.close();
}
void getRandomNumbers(int[] inputArr) throws IOException {
	Random rand = new Random();
	int number, count = 0;
	PrintWriter out = new PrintWriter(new File("Mrandom.txt"));
	while (count<size) {
		number = rand.nextInt(size) + 1;
		inputArr[count] = number;
		out.print(number + " ");
		count++;
	}
	out.close();
	System.out.println("The total numbers generated: " + count);
}

void doMergeSort(int lowerIndex, int higherIndex) {
	if (lowerIndex<higherIndex) {
		int middle = lowerIndex + (higherIndex - lowerIndex) / 2;
		// Below step sorts the left side of the array

		doMergeSort(lowerIndex, middle);
		// Below step sorts the right side of the array
		doMergeSort(middle+1, higherIndex);
		// Now merge both sides
		mergeParts(lowerIndex, middle, higherIndex);
	}
}
```

```
void mergeParts(int lowerIndex, int middle, int higherIndex) {
                for (int i = lowerIndex; i<= higherIndex; i++)
                        tempMergArr[i] = inputArr[i];
                int i = lowerIndex;
                int j = middle + 1;
                int k = lowerIndex;
                        while (i<= middle&&j<= higherIndex) {
                                if (tempMergArr[i] <= tempMergArr[j]) {
                                        inputArr[k] = tempMergArr[i];
                                        i++;
                                        cnt++;
                                } else {
                                        inputArr[k] = tempMergArr[j];
                                        j++;
                                        cnt++;
                                }
                                k++;
                        }
                while (i<= middle) {
                        inputArr[k] = tempMergArr[i];
                        k++;
                        i++;
                }
    while(j<=higherIndex)
    {
        inputArr[k] = tempMergArr[j];
                        k++;
                        j++;
    }
}
}

class Msort5 {
        public static void main(String args[]) throws IOException {
                MergeSort msort = new MergeSort();
                msort.process();        //call the merge sort process
        }
}
```

Output 1:

Enter the number of elements to sorted:(>5000):10000
The total numbers generated: 10000

The time complexity is ... 120568

Program 6

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

Knapsack Problem statement:

Given n objects, their weights wi , profits pi, where i=1 to n , and the capacity of the knapsack W, the objective is fill these objects in to the knapsack such that the maximum profit is earned and the weight of the objects chosen shouldn't exceed the capacity of the knapsack .

Dynamic Programming Method

```
import java.util.Scanner;
class Knapsack_DP
{
	static int max(int a, int b)
	{
		return (a>b)? a :b;
	}
	static int knapSack(int C, int wt[], int val[], int n)
	{
		int i, j;
		int [][] K = new int[n+1][C+1];
		// Build table K[][] in bottom up manner
		for (i = 0; i<= n; i++)
		{
			for (j = 0; j<= C; j++)
			{
				if (i==0 || j==0)
				K[i][j] = 0;
				else if (wt[i-1] <= j)
				K[i][j] = max(val[i-1]+K[i-1][j-wt[i-1]],K[i-1][j]);
				else
					K[i][j] = K[i-1][j];
			}
		}
		System.out.println("Knapscak table");
		for (i = 0; i<= n; i++)
		{
			for (j = 0; j<= C; j++)
			{
				System.out.print(K[i][j]+" ");
			}
			System.out.println();
		}
		System.out.print("Items Considered: ");
		i = n; j= C;
		while (i > 0 && j > 0)
		{
		 if (K[i][j] != K[i - 1][j])
		 {
			System.out.print(i + " ");
```

```
                        j -= wt[i - 1];
                    }
                    i -= 1;
                }
                return K[n][C];
        }
        public static void main(String args[])
        {
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter the number of Objects: ");
                int n = sc.nextInt();
                System.out.println("Enter the object's weights: ");
                int []wt = new int[n];
                for(int i=0; i<n; i++)
                        wt[i] = sc.nextInt();
                System.out.println("Enter the object's profits: ");
                int []val = new int[n];
                for(int i=0; i<n; i++)
                        val[i] = sc.nextInt();
                System.out.println("Enter the maximum capacity: ");
                int C = sc.nextInt();
                System.out.println("\nThe Maximum value that can be put in a knapsack of capacity W is: " +
knapSack(C, wt, val, n));
                sc.close();
        }
}
```

OUTPUT

```
Enter the number of Objects:
4
Enter the object's weights:
1 2 3 1
Enter the object's profits:
12 10 15 10
Enter the maximum capacity:
5
Knapscak table
0 0 0 0 0 0
0 12 12 12 12 12
0 12 12 22 22 22
0 12 12 22 27 27
0 12 22 22 32 37
Items Considered: 4 3 1
The Maximum value that can be put in a knapsack of capacity W is: 37
```

Greedy Method

```
import java.util.Scanner;
public class Knapsack
{
	double weight[];
	double benefit[];
	double ratio[];
	double W;
	Knapsack()
	{
		Scanner scan = new Scanner(System.in);
		System.out.println("Enter the number of objects: ");
		int nItems = scan.nextInt();
		weight = new double[nItems];
		ratio = new double[nItems];
		benefit= new double[nItems];
		System.out.println("Enter the object's weights");
		for (int i = 0; i<nItems; ++i)
			weight[i] = scan.nextDouble();
		System.out.println("Enter the object's profits");
		for (int i = 0; i<nItems; ++i)
			benefit[i] = scan.nextDouble();
		for (int i = 0; i<nItems; ++i)
			ratio[i] = benefit[i] / weight[i];
		System.out.println("Enter the Capacity of the knapsack: ");
		W = scan.nextDouble();
	}
	int getNext()
	{
		double highest = 0;
		int index = -1;
		for (int i = 0; i<benefit.length; ++i)
		{
			if (ratio[i] >highest) {
			highest = ratio[i];
			index = i;
			}
		}
		return index;
	}
	void fill()
	{
		double cW = 0; //current weight
		double cB = 0; //current benefit
		System.out.print("\n Objects  considered: ");
		while (cW<W)
		{
			int item = getNext();		//next highest ratio
			if (item == -1)
			{
				//No items left
```

```
                        break;
                    }
                    System.out.print((item+1)+" ");
                    if (cW + weight[item] <= W)
                    {
                        cW += weight[item];
                        cB += benefit[item];
                        //mark as used for the getNext() (ratio) function
                        ratio[item] = 0;
                    }
                    else
                    {
                        cB += (ratio[item] * (W - cW));
                        cW += (W - cW);
                        break;  //the knapsack is full
                    }
                }
                System.out.println("\nThe Optimal Solution i.e. Max Benefit = " + cB );
        }
        public static void main(String[] args)
        {
                Knapsack fk = new Knapsack();
                fk.fill();
        }
}
```

OUTPUT

```
Enter the number of objects:
3
Enter the object's weights
18 15 10
Enter the object's profits
25 24 18
Enter the Capacity of the knapsack:
20
Objects  considered: 3 2
The Optimal Solution i.e. Max Benefit = 34.0
```

Program 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.

```
import java.util.*;
class Dijikstras
{
        //declare distance array and cost matrix
        public int distance[] = new int[10];
        public int cost[][] = new int[10][10];
        public void compute(int n, int src)
```

```
{
        int flag[] = new int[n+1];
        int i, minpos=1,u,c,minimum;
        for(i=1;i<=n;i++)
        {
        flag[i]=0;
        //initial distance array which stores cost from source to all other nodes
        this.distance[i]=this.cost[src][i];
        }
        c=2;
                while(c<=n){
                        minimum=99;
                        //computes intermediate node u
                        for(u=1;u<=n;u++){
                                if(this.distance[u]<minimum && flag[u]!=1){
                                        minimum=this.distance[i];
                                        minpos=u;
                                }
                        }
                        flag[minpos]=1;
                        c++;
                    for(u=1;u<=n;u++)
                        {
                         if(this.distance[minpos]+this.cost[minpos][u] <this.distance[u] && flag[u]!=1 )
                                        this.distance[u] = this.distance[minpos] + this.cost[minpos][u];
                        }
                }
}
public static void main(String args[])
{
        int nodes,source,i,j;
        Scanner inp = new Scanner(System.in);
        //accept number of nodes of a graph
        System.out.println("Enter the Number of Nodes \n");
        nodes = inp.nextInt();
        //creates a class Dijikstras object by name d
        Dijikstras d = new Dijikstras();
        //accept cost matrix and sets 0 cost also to 999
        System.out.println("Enter the Cost Matrix Weights: \n");
        for(i=1;i<=nodes;i++)
                for(j=1;j<=nodes;j++) {
                        d.cost[i][j]=inp.nextInt();
                        if(d.cost[i][j]==0)
                        d.cost[i][j]=999;
                }
        System.out.println("Enter the Source Vertex :\n");
        source=inp.nextInt();
        //calls compute function to calculate shortest path from a
        //given source to all nodes
        d.compute(nodes,source);
```

```
                System.out.println("The Shortest Path from Source \t" + source + "\t to all other vertices are : \n");
                for(i=1;i<=nodes;i++)
                if(i!=source)
                        System.out.println("source :"+source+"\t destination :"+ i+"\t MinCost is :"+ d.distance[i]+"\t");
        }
}
```

Output 1

Enter the Number of Nodes

6

Enter the Cost Matrix Weights:

0 2 4 999 999 999

999 0 1 4 2 999
999 999 0 999 5 999
999 999 999 0 999 2
999 999 999 3 0 2
999 999 999 999 999 0

Enter the Source Vertex:

1

The Shortest Path from Source 1 to all other vertices are :

source :1 destination :2 MinCost is :2
source :1 destination :3 MinCost is :3
source :1 destination :4 MinCost is :6
source :1 destination :5 MinCost is :4
source :1 destination :6 MinCost is :6

Program 8

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

```
import java.util.Scanner;
class Compute
{
        // Number of vertices in the graph
        private static int nodes;
        int[ ][ ] graph;
        int[ ] parent;
        public void ReadFromConsole()
        {
                Scanner to_read = new Scanner(System.in);
                System.out.println("Enter number of nodes");
                nodes= to_read.nextInt();
                graph= new int [nodes+2][nodes+2] ;  // acquiring memory for matrix
                parent = new int[nodes+2];
```

```
        int  max= Integer.MAX_VALUE;
        System.out.println("Enter cost matrix");
        for(int i=1;i<=nodes;i++)
        {
                parent[i]=0;
                for(int j=1;j<=nodes;j++)
                {
                        graph[i][j] = to_read.nextInt();
                        if (graph[i][j]==0) graph[i][j] = max;
                }
        }
        to_read.close();
}

public void Kruskal()
{
        int i,j,a=0,b=0,u=0,v=0,min,mincost=0,ne=1;
        System.out.println("The edges of Minimum Cost Spanning Tree are");
        while(ne<nodes)
        {
                for(i=1,min=Integer.MAX_VALUE;i<=nodes;i++)
                for(j=1;j<= nodes;j++)
                if(graph[i][j] <min)
                {
                        min=graph[i][j];
                        a=u=i;
                        b=v=j;
                }
                u=find(u); v=find(v); if(uni(u,v)==1)
                {
                        System.out.println(ne++ + " edge(" + a + "," + b + ") = " + min);
                        mincost +=min;
                }
                graph[a][b]=graph[b][a]=Integer.MAX_VALUE;
        }
        System.out.println("Minimum cost = " + mincost);
}

public int find(int i)
{
        while(parent[i]!=0)
        i=parent[i];
        return i;
}
public int uni(int i,int j)
{
        if (i != j)
        {
                parent[j] = i;
                return 1;
        }
```

```
                return 0;
        }
}//End of class Compute
class Kruskals
{
        public static void main(String [] args)
        {
                Compute c = new Compute();
                c.ReadFromConsole();
                c.Kruskal();
        }
}
```

OUTPUT

```
Enter number of nodes

6
Enter cost matrix
0 3 0 0  6  5
3 0 1 0 0 4
0 1 0 6 0 4
0 0 6 0 8 5
6 0 0 8 0 2
5 4 4 5 2 0
The edges of Minimum Cost Spanning Tree are
1 edge(2,3) = 1
2 edge(5,6) = 2
3 edge(1,2) = 3
4 edge(2,6) = 4
5 edge(4,6) = 5
Minimum cost = 15
```

Program 9

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

```
import java.util.Scanner;
class MST
{
        // Number of vertices in the graph
        private static int Vertices;
        int[][] graph;

        public void ReadFromConsole()
        {
                Scanner to_read = new Scanner(System.in);
                System.out.println("Enter number of nodes");
                Vertices= to_read.nextInt();

                graph = new int[Vertices][Vertices] ;  // acquiring memory for matrix
```

```
            int v = Integer.MAX_VALUE;
            System.out.println("Enter Cost Matrix");
            for(int i=0;i<Vertices;i++)
            for(int j=0;j<Vertices;j++)
       {
              graph[i][j] = to_read.nextInt();
              if (graph[i][j]==0) graph[i][j] = v;
       }
            to_read.close();
     }
    // A utility function to print the constructed MST stored in
    // parent[]
    void printMST(int parent[], int n)
    {

            int sum=0;
            System.out.println("Edge   Weight");
            for (int i = 1; i< Vertices; i++)
            {
                    sum = sum + graph[i][parent[i]];
                    System.out.println(parent[i]+" - "+ i+"    "+ graph[i][parent[i]]);
            }
            System.out.println("Weight of Spanning tree is " + sum);
    }
    // A utility function to find the vertex with minimum key
    // value, from the set of vertices not yet included in MST
    int minKey(int key[], int mstSet[])
    {
            // Initialize min value
            int min = Integer.MAX_VALUE, min_index=-1;

            for (int v = 0; v < Vertices; v++)
                    if (mstSet[v] == 0 && key[v] < min)
                    {
                            min = key[v];
                            min_index = v;
                    }
            return min_index;
    }
    void primMST()
    {
            // To represent set of vertices not yet included in MST
            //Boolean mstSet[] = new Boolean[V];
            int mstSet[] = new int[Vertices];
            // Array to store constructed MST
            int parent[] = new int[Vertices];
// Key values used to pick minimum weight edge in cut
            int key[] = new int [Vertices];
            // Initialize all keys as INFINITE
            for (int i = 0; i< Vertices; i++)
            {
```

```
                key[i] = Integer.MAX_VALUE;
                mstSet[i] = 0;
            }
            // Always include first 1st vertex in MST.
            key[0] = 0;     // Make key 0 so that this vertex is
            // picked as first vertex
            parent[0] = -1; // First node is always root of MST
            for (int count = 0; count < Vertices-1; count++)
            {
                // Pick the minimum key vertex from the set of vertices not yet in MST
                int u = minKey(key, mstSet);
                // Add the picked vertex to the MST Set
                mstSet[u] = 1;
                for (int v = 0; v < Vertices; v++)
                if (graph[u][v]!=0 &&mstSet[v] == 0   &&   graph[u][v] <  key[v])
                {
                    parent[v]  = u;
                    key[v] = graph[u][v];
                }
            }
            // print the constructed MST
            printMST(parent, Vertices);
        }
}
class Prims
{
    public static void main(String[] args)
    {
        MST c = new MST();
        c.ReadFromConsole();
        c.primMST();
    }
}
```

Output 1

```
Enter number of nodes
6
Enter Cost Matrix
0 3 0 0  6  5
3 0 1 0 0 4
0 1 0 6 0 4
0 0 6 0 8 5
6 0 0 8 0 2
5 4 4 5 2 0
Edge   Weight
0 - 1    3
1 - 2    1
5 - 3    5
5 - 4    2
1 - 5    4
```

Weight of Spanning tree is 15

Program 10
Write Java programs to
(a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm.

```
import java.util.Scanner;

public class Floyd
{
   private int dmat[][];
   private int n;
   public static final int INFINITY = 999;

   public Floyd(int n)
   {
         dmat = new int[n + 1][n + 1];
         this.n = n;
   }

   public void floyd_compute(int adjmat[][])
   {
      for (int i = 1; i<= n; i++)
      {
         for (int j = 1; j <= n; j++)
         {
                  dmat[i][j] = adjmat[i][j];
         }
      }

      for (int k = 1; k <= n; k++)
      {
         for (int i = 1; i<= n; i++)
         {
            for (int j = 1; j <= n; j++)
            {
               if (dmat[i][k] + dmat[k][j] <dmat[i][j])
                        dmat[i][j] = dmat[i][k] + dmat[k][j];
            }
         }
      }

      for (int i = 1; i<= n; i++)
               System.out.print("\t" + i);
         System.out.println();
      for (int i = 1; i<= n; i++)
      {
               System.out.print(i + "\t");
               for (int j = 1; j <= n; j++)
                 {
                        System.out.print(dmat[i][j] + "\t");
```

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

    public static void main(String[] arg)
    {
        int adjmat[][];
        int n;

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter the number of vertices");
        n = scan.nextInt();
        adjmat = new int[n + 1][n + 1];
        System.out.println("Enter the Weighted Matrix for the graph");
        for (int i = 1; i<= n; i++)
        {
            for (int j = 1; j <= n; j++)
            {
                    adjmat[i][j] = scan.nextInt();
                if (i == j)
                {
                    adjmat[i][j] = 0;
                   continue;
                }
                if (adjmat[i][j] == 0)
                {
                    adjmat[i][j] = INFINITY;
                }
            }
        }
System.out.println("The All Pairs Shortest Distance of the Graph is: ");
Floyd floydobj = new Floyd(n);
floydobj.floyd_compute(adjmat);
scan.close();
    }
}
```

```
Enter the number of vertices
4
Enter the Weighted Matrix for the graph
0 999 3 999
2 0 999 999
999 7 0 1
6 999 999 0
The All Pairs Shortest Distance of the Graph is:
        1       2       3       4
1       0       10      3       4
2       2       0       5       6
3       7       7       0       1
4       6       16      9       0
```

10b. Implement Travelling Sales Person problem using Dynamic programming.

```
import java.util.Scanner;
class TSP {
	static int cost = 0;

	public static void main(String[] args) {

		int a[][] = new int[10][10];
		int visited[] = new int[10];
		int n;
		System.out.println("Enter number of cities: ");
		Scanner s = new Scanner(System.in);
		n = s.nextInt();
		create(a,visited,n);
		System.out.println("\n\npath is");
		mincost(a,n,0,visited);
		display();

	}

	public static void display() {
		// TODO Auto-generated method stub
		System.out.println("\n total cost of tour=" + cost);
	}

	public static void mincost(int[][] a, int n, int city, int[] visited) {

		int i,city_no;
		visited[city]=1;
		System.out.print((city+1)+"--> ");
		city_no = least(a,visited,n,city);
		if(city_no==999){
			city_no=0;
			System.out.print(" "+(city_no+1));
			cost+=a[city][city_no];
			return;
		}
		mincost(a,n,city_no,visited);

	}

	public static int least(int[][] a, int[] visited, int n, int c)
//returns the city which is reachable from source in minimum cost
{
		int i,min_node=999;
		int min=999;
		int new_min=0;
		for(i=0;i<n;i++){
```

```
        if((a[c][i]!=0)&&visited[i]==0)//if path  exist and next city to reach is not yet visited (minimum distance city nerer to source i.e 0)
                        if(a[c][i]<min){
                                min = a[i][0]+a[c][i];
                                new_min = a[c][i];
                                min_node = i;
                        }
                }
                if(min!=999)
                        cost+=new_min;
                return min_node;
        }

        public static void create(int[][] a, int[] visited, int n)
       {
                System.out.println("\n enter the cost matrix:");
                for(int i=0;i<n;i++){
                        System.out.println("row:" + (i+1));
                        for(int j=0;j<n;j++)
                        {
                                Scanner scan = new Scanner(System.in);
                                a[i][j] = scan.nextInt();
                        }
                        visited[i]=0;
                }
                System.out.println("\n\nThe cost matrix");
                        for(int i=0;i<n;i++){
                                System.out.println("\n");
                                for(int j=0;j<n;j++){
                                        System.out.print(" "+a[i][j]);

                                }
                        }
                }
        }
```

```
Enter number of cities:
4

 enter the cost matrix:
row:1
0
10
15
20
row:2
5
0
9
10
```

```
row:3
6
13
0
12
row:4
8
8
9
0
```

The cost matrix

```
 0 10 15 20

 5 0 9 10

 6 13 0 12

 8 8 9 0
```

path is
1--> 2--> 4--> 3--> 1
total cost of tour=35

Program 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.

```
import java.util.Scanner;
public class SumOfSubsets
{
      int[ ] w;
      int[ ] x;
     int sum;
     int total = 0;

   public void process()
  {
        getData();
        System.out.println("The subsets are:");
        subset(0, 1, total);
   }
   private void getData()
  {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of elements:");
```

```
        int n = sc.nextInt();
        w = new int[n + 1];
        x = new int[n + 1];
        System.out.println("Enter " + n + " Elements :");
        for (int i = 1; i< n + 1; i++)
        {
            w[i] = sc.nextInt();
            total += w[i];
        }
        System.out.println("Enter the sum to be obtained: ");
        sum = sc.nextInt();
        if (total < sum)
       {
            System.out.println("No possible subsets!!!");
            System.exit(1);
        }
            sc.close();
    }
    private void subset(int s, int k, int r)
   {
        int i = 0;
        x[k] = 1;

        if (s + w[k] == sum)
        {
            System.out.print("(");
            for (i = 1; i<= k; i++)
            {
                    if(x[i]==1)
                            System.out.print(" " + w[i]);
            }
            System.out.println(" )");
        }
        else if ((s + w[k] + w[k + 1]) <= sum)
        {
            subset(s + w[k], k + 1, r - w[k]);
        }
        if ((s + r - w[k]) >= sum && (s + w[k + 1]) <= sum)
        {
            x[k] = 0;
                    subset(s, k + 1, r - w[k]);
        }
    }
    public static void main(String[] args) {
        new SumOfSubsets().process();
    }
}
```

```
Enter the number of elements:5
Enter 5 Elements :
1 2 5 6 8
```

```
Enter the sum to be obtained:
9
The subsets are:
( 1 2 6 )
( 1 8 )

Enter the number of elements:5
Enter 5 Elements :
1 2 3 4 5
Enter the sum to be obtained:
16
No possible subsets!!!
```

Program 12

Design and implement in Java to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking principle

```
import java.util.*;
public class Hamiltonian {
static int MAX=25;
static int vertex[]=new int[MAX];
public static void main(String[] args) {
        int i,j,v1,v2,edges,n;
        int G[][]=new int[MAX][MAX];
        System.out.println("\n\t program for Hamiltonian cycle");
        System.out.println("enter the number of vertices of graph");
        Scanner in =new Scanner(System.in);
        n=in.nextInt();
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        G[i][j]=0;
        vertex[i]=0;
        }
        }
        System.out.println("\n enter the total number of edges:");
        edges=in.nextInt();
        for(i=1;i<=edges;i++)
        {
                System.out.println("enter the edge:");
                v1=in.nextInt();
                v2=in.nextInt();
                G[v1][v2]=1;
                G[v2][v1]=1;
        }
        vertex[1]=1;
        System.out.println("Hamiltonian cycle");
        H_cycle(G,n,2);
}
public static void next_vertex(int G[][],int n,int k)
{
```

```
        int j;
        while(true)
        {
                vertex[k]=(vertex[k]+1)%(n+1);
                if(vertex[k]==0)
                        return;
                if(G[vertex[k-1]][vertex[k]]!=0)
                {
                        for(j=1;j<=k-1;j++)
                        {
                                if(vertex[j]==vertex[k])
                                        break;
                        }
                        if(j==k)
                        {
                                if((k<n)||((k==n) && (G[vertex[n]][vertex[1]]!=0)))
                                        return;
                        }
                }
        }
}
public static void H_cycle(int G[][],int n,int k)
{
        int i;
        while(true)
        {
                next_vertex(G,n,k);
                if(vertex[k]==0)
                        return;
                if(k==n){
                        System.out.println("\n");
                        for(i=1;i<=n;i++)
                                System.out.print(vertex[i]+"-->");
                        System.out.println(" "+vertex[1]);
                }
                else
                        H_cycle(G,n,k+1);
        }
}
}
```

output

```
        program for Hamiltonian cycle
enter the number of vertices of graph
4
enter the total number of edges:
5
enter the edge:
1 2
enter the edge:
1 3
```

enter the edge:
1 4
enter the edge:
2 3
enter the edge:
3 4
Hamiltonian cycle

1-->2-->3-->4--> 1

1-->4-->3-->2--> 1

Output:-

enter the number of vertices of graph
8
enter the total number of edges:
11
enter the edge:
1 2
enter the edge:
1 3
enter the edge:
1 7
enter the edge:
2 3
enter the edge:
2 8
enter the edge:
3 4
enter the edge:
3 6
enter the edge:
4 5
enter the edge:
5 6
enter the edge:
6 7
enter the edge:
7 8
Hamiltonian cycle

1-->2-->8-->7-->6-->5-->4-->3--> 1

1-->3-->4-->5-->6-->7-->8-->2--> 1