JAVA JOURNAL 4TH SEMESTER

usn: 2JR18CS051
by: PRAVEEN UKKOJI
dept: COMPUTER SCIENCE
div: A

Experiment No. 1

A) Create a java class called **student** with the following details as variables within it.

1. USN
2. Name
3. Branch
4. Phone

Write a java program to create n student objects and print the USN, Name, Branch and Phone of these objects with suitable headings.

Algorithm:

1. Class **student** is created
2. Declare variables USN, Name, Branch, and Phone no.
3. A constructor of **student** class is created to initialize these variables
4. Function **display** is created that prints these details like usn, name, branch and phone no
5. Multiple objects of **student** class calls the function **display** to print the details contained in **student** class.

Source code:

```
import java.utils.*;
public class student {
   String USN;
   String Name;
   String Branch;
   long Phone;

   void insert(String usn, String name, String branch, long phone) {
      this.USN = usn;
      this.Name= name;
      this.Branch = branch;
      this.Phone = phone;
      }
   void display() {
      System.out.println(USN + " " + Name + " " + Branch + " " + Phone);
   }
   public static void main(String[] args) {
      student s[];
      s = new student[100];
      Scanner in = new Scanner(System.in);
```

```
        System.out.print("Enter no of students: ");
        int n = in.nextInt();
        for(int i=0;i<n;i++){
            s[i] = new student();
        }
        for(int i=0;i<n;i++) {
            System.out.println("Enter details:");
            System.out.print("Enter USN:");
            String usn = in.next();
            System.out.print("Enter Name:");
            String name = in.next();
            System.out.print("Enter Branch:");
            String branch = in.next();
            System.out.print("Enter Phone:");
            long phone = in.nextLong();
            s[i].insert(usn, name, branch, phone);
        }
        for(int i=0;i<n;i++) {
            s[i].display();
            System.out.println("");
        }
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro prg 1 % javac student.java
praveenukkoji@Praveens-MacBook-Pro prg 1 % java student
Enter no of students: 2
Enter details:
Enter USN:2jr18cs051
Enter Name:praveen
Enter Branch:cs
Enter Phone:9686391577
Enter details:
Enter USN:2kl15cs051
Enter Name:prem
Enter Branch:cs
Enter Phone:1234567890
2jr18cs051 praveen cs 9686391577

2kl15cs051 prem cs 1234567890

praveenukkoji@Praveens-MacBook-Pro prg 1 %
```

B) Write a java program to implement the stack using arrays. Write push(), pop() and display() methods to demonstrate its working.

Algorithm:

1. Class created and variables are defined like array[], top, size.
2. A customized constructor of same class is used for initializing size, top variables and the array[].
3. A function created for pushing the elements into stack :

```
push( int eletopush)
{
    if( stack is not full)
    {
        top++;
        array[top] = eletopush;
    }
    else
    {
        stack is full;
    }
}
```

4. A function created for poping the elements into stack :

```
pop( int eletopop)
{
        if( stack is not full)
{
        ele = array[top];
        top--;
}
else
{
        stack is full;
}
}
```

5. A function is created for displaying the elements in the stack:

```
printelments()
{
        if(top >= 0)
        {
                print all elements;
        }
```

```
        else
        {
            stack is empty;
        }
    }
```

6. A Boolean function is created to check whether stack is empty or full :

```
    Boolean isfull()
    {
        return (top == size-1);
    }
    Boolean isempty()
    {
        return top ==  -1;
    }
```

Source code:

```
import java.utils.*;
public class stack {
   static int stack[], top = -1;
   public static void main(String[] args) {
      System.out.print("Enter Stack Size: ");
      Scanner in = new Scanner(System.in);
      int size = in.nextInt();
      stack = new int[size];
      System.out.println("\nOpEons:\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT");
      System.out.print("Enter your Choice: ");
      int choice = in.nextInt();
      while(choice != 4)
      {
         if(choice == 1) {
            System.out.print("\n Enter Element to push: ");
            int element = in.nextInt();
            if(top == size-1)
               System.out.println("\n Stack is full.");
            else
                stack[++top] = element;
         }
         else if(choice==2) {
            if(top == -1)
               System.out.println("\n Stack is empty.”);
            else
               System.out.println("\n Popped element is: "+ stack[top--]);
```

```
        }
        else if(choice==3) {
            if(top == -1)
                System.out.println("\n Empty stack.");
            else {
                System.out.print("\n Stack Elements are:
                "); for(int i=top; i>=0 ;i--)
                    System.out.print(stack[i] + " ");
            }
        }
        else
            System.out.println("\n Enter Correct Choice \n");
            System.out.println("\nOpEons:\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT");
            System.out.print("Enter your Choice: "); choice = in.nextInt();
    }
    in.close();
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro prg 1 % javac stack.java
praveenukkoji@Praveens-MacBook-Pro prg 1 % java stack
Enter Stack Size: 10

Options:
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your Choice: 1

Enter Element to push: 10

Options:
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your Choice: 1

Enter Element to push: 20

Options:
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your Choice: 3

Stack Elements are: 20 10
Options:
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your Choice: 2

Popped element is: 20

Options:
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your Choice: 4
praveenukkoji@Praveens-MacBook-Pro prg 1 %
```

Experiment No. 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.

Algorithm:

1. Class **staff** is created.
2. Variables are declared in class like StaffId, name, phoneno, salary.
3. Constructor is used to initialize the values of these variables.
4. Subclass "**teaching**" is created that inherits superclass "**staff**".
5. A class is created "**technical**" that inherits super class "**staff**".
6. A class is created named "**contract**" that inherits the super class "**staff**"
7. From the main method, objects are made of each subclass and functions are called.

Source code:

```
import java.utils.*;


class Staff {
   int StaffId;
   String Phone;
   int Salary;
   String Name;


   public Staff(int staffId, String phone, int salary, String name) {
      this.StaffId = staffId;
      this.Phone = phone;
      this.Salary = salary;
      this.Name = name;
   }


   void display() {
      System.out.println("");
      System.out.println("Staff Id: " + StaffId);
      System.out.println("Phone: " + Phone);
      System.out.println("Salary: " + Salary);
      System.out.println("Name: " + Name);
   }
}
```

```
class Teaching extends Staff {
    String Domain;
    int No_of_publications;
    public Teaching(int staffId, String phone, int salary, String name, String
domain, int no_of_ publications) {
        super(staffId, phone, salary, name);
        this.Domain = domain;
        this.No_of_ publications = no_of_ publications;
    }

    void TeachingDisplay() {
        System.out.println("");
        System.out.println("Teaching Staff Details:");
        super.display();

        System.out.println("Domain: " + Domain);
        System.out.println("No_of_ publications: " + No_of_ publications);
    }
}

class Technical extends Staff {
    String Skills;
    public Technical(int staffId, String phone, int salary, String name, String skill)
        { super(staffId, phone, salary, name);
        this.Skills = skill;
    }
```

```
    void TechnicalDisplay() {
        System.out.println("");
        System.out.println("Technical Staff Details:");
        super.display();
        System.out.println("Skills: " + Skills);
    }
}

class Contract extends Staff {
    int Period;
    public Contract(int staffId, String phone, int salary, String name, int period)
        {
        super(staffId, phone, salary, name);
        this.Period = period;

    }

    void ContractDisplay() {
        System.out.println("");
        System.out.println("Contract Staff Details:");
        super.display();
        System.out.println("Period: " + Period + " years");
    }
}

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

```
        Teaching Te1 = new Teaching(11, "9987654341", 300000, "Praveen", "Cse",
        10);
        Teaching Te2 = new Teaching(12, "9987654341", 31500, "Pavya", "Mech", 11);
        Teaching Te3 = new Teaching(13, "9987654341", 3000, "Pravin", "EE", 12);

        Te1.TeachingDisplay();
        Te2.TeachingDisplay();
        Te3.TeachingDisplay();

        Technical Tn1 = new Technical(11, "9987654341", 22000, "Pavya", "C");
        Technical Tn2 = new Technical(12, "9987654341", 23000, "Pavya2", "java");
        Technical Tn3 = new Technical(13, "9987654341", 23000, "Pavya3", "C++");

        Tn1.TechnicalDisplay();
        Tn2.TechnicalDisplay();
        Tn3.TechnicalDisplay();

        Contract C1 = new Contract(11, "9987654341", 35000, "Pavya1", 3);
        Contract C2 = new Contract(12, "9987654341", 36000, "Pavya2", 2);
        Contract C3 = new Contract(13, "9987654341", 37000, "Pavya3", 1);

        C1.ContractDisplay();
        C2.ContractDisplay();
        C3.ContractDisplay();

    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 2 % javac main_staff.java
praveenukkoji@Praveens-MacBook-Pro termwork 2 % java main_staff

Teaching Staff Details:

Staff Id: 11
Phone: 9987654341
Salary: 300000
Name: Praveen
Domain: Cse
No_of_publications: 10

Teaching Staff Details:

Staff Id: 12
Phone: 9987654341
Salary: 31500
Name: Pavya
Domain: Mech
No_of_publications: 11

Teaching Staff Details:

Staff Id: 13
Phone: 9987654341
Salary: 3000
Name: Pravin
Domain: EE
No_of_publications: 12

Technical Staff Details:

Staff Id: 11
Phone: 9987654341
Salary: 22000
Name: Pavya
Skills: C

Technical Staff Details:

Staff Id: 12
Phone: 9987654341
Salary: 23000
Name: Pavya2
Skills: java

Technical Staff Details:

Staff Id: 13
Phone: 9987654341
Salary: 23000
Name: Pavya3
Skills: C++
```

```
Contract Staff Details:

Staff Id: 11
Phone: 9987654341
Salary: 35000
Name: Pavya1
Period: 3 years

Contract Staff Details:

Staff Id: 12
Phone: 9987654341
Salary: 36000
Name: Pavya2
Period: 2 years

Contract Staff Details:

Staff Id: 13
Phone: 9987654341
Salary: 37000
Name: Pavya3
Period: 1 years
praveenukkoji@Praveens-MacBook-Pro termwork 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 < name, dd, mm, yyyy > using StringTokenizer class considering the delimiter character as "/".

Algorithm :

1. Class named "**customer**" is created.
2. Variables are defined as name and date_of_birth
3. A object is created of StringTokenizer class and is used for calling the method of this class.

Source code:

```
import java.utils.Scanner;
import java.utils.StringTokenizer;

public class customer {
   String Name ;
   String Date;
   String Month;
   String Year;
   void read() {
      Scanner in = new Scanner(System.in);
      System.out.println("\n Enter Name and DOB in Name, DD/MM/YYYY Format:
");
      String str = in.next();
      StringTokenizer st = new StringTokenizer(str, "," + "/");
         this.Name=st.nextToken();
         this.Date=st.nextToken();
         this.Month=st.nextToken();
         this.Year=st.nextToken();
         this.Name=this.Name.trim();
         this.Date=this.Date.trim();
         this.Month=this.Month.trim();
         this.Year=this.Year.trim();

      in.close();
   }
```

```
    void display() {
        System.out.println("\n Customer Details is: ");
        System.out.println(this.Name + "," + this.Date + "," + this.Month + ","
+ this.Year);
    }

    public static void main(String[] args) {
        customer c1 = new customer();
        c1.read();
        c1.display();
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 2 % javac customer.java
praveenukkoji@Praveens-MacBook-Pro termwork 2 % java customer

Enter Name and DOB in Name, DD/MM/YYYY Format:
PRAVEEN,20/09/2000

Customer Details is:
PRAVEEN,20,09,2000
praveenukkoji@Praveens-MacBook-Pro termwork 2 % #
```

Experiment No. 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.

Algorithm:

1. A class is created containing the main method
2. Two variables are declared i.e. a and b
3. Input is obtained from console

   ```
   Scanner sc = new Scanner(System.in); a =
   sc.nextInt();
   b = sc.nextInt();
   ```

4. The code to calculate division is kept under try block

   ```
   try{
       System.out.println(a/b);
   }
   ```

5. The arithmetic exception raised when b=0 is handled in catch block that follows try block

   ```
   catch(ArithmeticException e){

           e.printStackTrace();
   }
   ```

Source code:

```
import java.utils.Scanner;
public class Excep {


   public static void main(String[] args) {
     Scanner in = new Scanner(System.in);
     int a,b,c;
     System.out.println("Enter val of a and b: ");
     a = in.nextInt();
     b = in.nextInt();
     try {
          c = a/b;
          System.out.println("c = " + c);
     }
     catch(ArithmeticException e) {
          System.out.println("Exception Encountered is "+ e);
     }
      in.close();
   }
}
```

Output :

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.

Algorithm:

1. Three classes first, second and third are created
2. Class first generates an integer using random number generator and prints the integer with thread t1.
3. Next class second is called to generate square of the number and print it with thread t2.
4. At last class third is called to generate cube of the number and print it with thread t3.

Source code :

```
import java.utils.*;
class second implements Runnable
{
   public int x;
   public second (int x)
   {
      this.x=x;
   }
      public void run()
   {
      System.out.println("Second thread : Square of the number is: " + x * x);
   }
}

class third implements Runnable
{
   public int x;
      public third(int x)
      {
         this.x=x;
      }
```

```
    public void run()
    {
        System.out.println("third thread : Cube of the number is: " + x * x *
        x); System.out.println("");
    }
}

class first extends Thread
{
    public void run()
    {
        int num=0;
        Random r=new Random();
        try
        {
            for(int i=0;i<5;i++)
            {
                num=r.nextInt(100);
                System.out.println("first thread generated number is: " +
                num); Thread t2=new Thread (new second(num)); t2.start();

                Thread t3=new Thread(new third(num));
                t3.start();
                Thread.sleep(1000);
            }
        }
```

```
        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
}

public class prg3b
{
    public static void main(String args[])
    {
        first a=new first();
        a.start();
    }
}
```

Output :

```
praveenukkoji@Praveens-MacBook-Pro termwork 3 % javac prg3b.java
praveenukkoji@Praveens-MacBook-Pro termwork 3 % java prg3b
first thread generated number is: 52
Second thread : Square of the number is: 2704
third thread : Cube of the number is: 140608

first thread generated number is: 88
Second thread : Square of the number is: 7744
third thread : Cube of the number is: 681472

first thread generated number is: 47
Second thread : Square of the number is: 2209
third thread : Cube of the number is: 103823

first thread generated number is: 38
Second thread : Square of the number is: 1444
third thread : Cube of the number is: 54872

first thread generated number is: 36
Second thread : Square of the number is: 1296
third thread : Cube of the number is: 46656

praveenukkoji@Praveens-MacBook-Pro termwork 3 %
```

Experiment No. 4

Sort a given set of n integers elements using 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 v/s 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 methods works along with its time complexity analysis worst case, average case and best case.

Algorithm:

Quick sort (A[l....r])

// Sorts a sub array by recursive quick sort
//Input : A sub array A[l..r] of A[0..n-1] ,defined by its left and right indices l //and r
// Output : The sub array A[l..r] sorted in non-decreasing order
Steps:
if l < r
s = Partition (A[l..r]) //s is a split position
Quick sort (A[l ...s-1])
Quick sort (A[s+1...r])

Partition (A[l...r])
//Partition function divides an array into sub arrays by using its first element as pivot
// Input : A sub array A[l...r] of A[0...n-1] defined by its left and right indices l and r (l < r) // Output : A partition of A[l...r], with the split position returned as this function's value

Steps:
p=A[l]
i=l;
j=r+1; repeat
repeat i= i+1 until A[i] >= p repeat j=j-1 until A[J] <= p Swap (A[i],A[j])

until I >=j
Swap (A[i],A[j]) // Undo last Swap when i>= j Swap (A[l],A[j])
Return j

Source code :

```
import java.utils.Scanner;

public class quick {
   static int a[];

   static void qsort(int leu, int right)
   {
      if(leu<right)
      {
         int p = partition(leu, right);
         qsort(leu, p-1);
         qsort(p+1, right);
      }

}
   static int partition (int low, int high)
   {
      int pivot = a[high];
      int pindex=low;
      for(int i=low;i<=high-1;i++)
      {
```

```
        if(a[i]<=pivot)
        {
            int temp = a[pindex];
            a[pindex] = a[i];
            a[i] = temp;

            pindex++;
        }
    }

    int temp = a[pindex];
    a[pindex] = a[high];
    a[high] = temp;

    return pindex;
}
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    int n, i;
    System.out.println("Enter the no. of elements: ");
    n = s.nextInt();
    a = new int[n];
    System.out.println("The Random Numbers are:");
```

```
        for (i = 0; i < n; i++)
        {
            a[i] = (int) (Math.random() * 100);
            System.out.print(a[i]+" ");
        }
        long startTime = System.nanoTime();
        qsort(0, n-1);
        long endTime = System.nanoTime();
        long duration = (endTime - startTime);

        System.out.println("\n The Sorted Numbers are: ");
        for (i = 0; i < n; i++) {
            System.out.print(a[i] +" ");
        }
        System.out.println("\n Time taken is: "+ duration/1000 + " microseconds" );
    }
}
```

Output :

```
praveenukkoji@Praveens-MacBook-Pro termwork 4 % javac quick.java
praveenukkoji@Praveens-MacBook-Pro termwork 4 % java quick
Enter the no. of elements:
5
The Random Numbers are:
0 13 27 63 48
The Sorted Numbers are:
0  13  27  48  63
Time taken is: 13 microseconds
praveenukkoji@Praveens-MacBook-Pro termwork 4 % java quick
Enter the no. of elements:
7
The Random Numbers are:
14 95 52 49 24 49 84
The Sorted Numbers are:
14  24  49  49  52  84  95
Time taken is: 13 microseconds
praveenukkoji@Praveens-MacBook-Pro termwork 4 %
```

Experiment No. 5

Sort a given set of n integers 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 v/s 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 methods works along with its time complexity analysis worst case, average case and best case.

Algorithm:

Merge sort (A[0...n-1]

// Sorts array A[0..n-1] by Recursive merge sort

// Input : An array A[0..n-1] elements

// Output : Array A[0..n-1] sorted in non-decreasing order

If n > 1

Copy A[0...(n/2)-1] to B[0...(n/2)-1]

Copy A[n/2...n-1] to C[0...(n/2)-1]

Mergesort (B[0...(n/2)-1])
Mergesort (C[0...(n/2)-1])

Merge(B,C,A)

Merge (B[0...p-1], C[0...q-1],A[0....p+q-1])

// merges two sorted arrays into one sorted array

// Input : Arrays B[0..p-1] and C[0...q-1] both sorted

// Output : Sorted array A[0.... p+q-1] of the elements of B and C i = 0;

```
j = 0;
k= 0;
while i < p and j < q do

if B[i] <= C[j]

A[k]= B[i];       i++;
else
A[k] = C[j];      j = j+1;
k=k+1;
if i = = p
    Copy C[ j..q-1] to A[k....p+q-1]

else

    Copy B[i ... p-1] to A[k ...p+q-1]
```

Source code :

```
import java.utils.Random;
import java.utils.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]+" ");

System.out.println("");

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("Time taken to sort array is: "+elapseTime+" nano
seconds.");

System.out.println("Sorted array is");
```

```
        for( i=0;i<n;i++)
            System.out.print(array[i]+" ");
        System.out.println("");
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 5 % javac mergesort.java
praveenukkoji@Praveens-MacBook-Pro termwork 5 % java mergesort
Enter the array size
8
Array before sorting:
19 12 10 3 3 11 15 13
Time taken to sort array is: 403647 nano seconds.
Sorted array is
3 3 10 11 12 13 15 19
praveenukkoji@Praveens-MacBook-Pro termwork 5 %
```

Experiment No. 6

Implement in JAVA, the 0/1 knapsack problem using

A) Dynamic programming method.

Algorithm:
// Initialization of first column and first row elements

• Repeat for i = 0 to n
set V(i,0) = 0
• Repeat for j = 0 to W Set V(0,j) = 0
//complete remaining entries row by row • Repeat for i = 1 to n
repeat for j = 1 to W

if(w_i <=j)V(i,j))=max{V(i-1,j),V(i-1,j-w_i)+v_i }if(w_i >j) V(i,j) = V(i-1,j)
• Print V(n,W)

Source code:

```
import java.utils.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: "+ 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.print("\n Items selected : ");

    for ( i = 1; i < N + 1; i++)
      if (selected[i] == 1)

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

Output:

```
praveenukkoji@Praveens-MacBook-Pro tremwork 6 % javac knapsackDP.java
praveenukkoji@Praveens-MacBook-Pro tremwork 6 % java knapsackDP
Enter number of elements
4

Enter weight for 4 elements
10
6
12
5

Enter value for 4 elements
2
3
6
7

Enter knapsack weight
9


The optimal solutionis: 7

Items selected : 4
praveenukkoji@Praveens-MacBook-Pro tremwork 6 %
```

B) Greedy Method

Algorithm:

- Assume knapsack holds weight W and items have value v_i and weight w_i
- Rank items by value/weight ratio: v_i / w_i

  Thus: $v_i / w_i \geq v_j / w_j$, for all $i \leq j$

- Consider items in order of decreasing ratio
- Take as much of each item as possible based on knapsack's capacity.

Source code:

```
import java.utils.Scanner;

public class knapsackGD {
   static int size=1000,n,j;
   static float max, sum=0;
   void knap(int W, int wei[],int val[]) {
      while(W>=0){
         max=0;
         for(int i=0;i<n;i++) {
            if((float)val[i]/(float) wei[i]>max){
               max=((float)val[i]/(float)wei[i]);
                j=i;
               }
            }
         if(wei[j]>W){
            System.out.println("Weight added to knapsack "+W);
            sum+=W*max;
            W=-1;
         }
```

```
        else{
            System.out.println("Weigth added to knapsack "+wei[j]);
            sum+=(float)val[j];
            W=W-wei[j];
            val[j]=0;
        }
    }
    System.out.println("The Maximum Value/Profit is "+sum );
}
public static void main(String[] args) {
    System.out.println("-------Knapsack Problem ------
    "); System.out.println("Enter weight of knapsack ");
    Scanner sc = new Scanner(System.in);

    int W=sc.nextInt();
    int w[]=new int[size];
    int v[]=new int[size];
    System.out.println("Enter number of items ");
    n=sc.nextInt();
    System.out.println("Enter Weights "+n+" Respective Items");
    for(int i=0;i<n;i++) {
        w[i]=sc.nextInt();
    }
```

```
        System.out.println("Enter value of "+n+" Item ");
        for(int i=0;i<n;i++) {
            v[i]=sc.nextInt();
        }
        knapsackGD k = new knapsackGD();
        k.knap(W, w, v);
        sc.close();
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro tremwork 6 % javac knapsackGD.java
praveenukkoji@Praveens-MacBook-Pro tremwork 6 % java knapsackGD
-------Knapsack Problem ------
Enter weight of knapsack
9
Enter number of items
4
Enter Weights 4 Respective Items
10
6
12
5
Enter value of 4 Item
2
3
6
7
Weigth added to knapsack 5
Weight added to knapsack 4
The Maximum Value/Profit is 9.0
praveenukkoji@Praveens-MacBook-Pro tremwork 6 %
```

Experiment No. 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.

Algorithm:

1. Read number of vertices of graph G .
2. Read weighted graph G .
3. Print weighted graph .
4. Initialize distance from source for all vertices as weight between source node and other vertices, i, and none in tree .

Source code:

```
import java.utils.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:

```
praveenukkoji@Praveens-MacBook-Pro termwork 7 % javac Dijkstra.java
praveenukkoji@Praveens-MacBook-Pro termwork 7 % java Dijkstra
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
praveenukkoji@Praveens-MacBook-Pro termwork 7 %
```

Experiment No. 8

Find Minimum Cost Spanning Tree of a given undirected graph using

A) **Kruskal's** Algorithm

Algorithm:

Start with an empty set A, and select at every stage the shortest edge that has not been chosen or rejected, regardless of where this edge is situated in graph.

- Initially, each vertex is in its own tree in forest.
- Then, algorithm consider each edge in turn, order by increasing weight.
- If an edge (u, v) connects two different trees, then (u, v) is added to the set of edges of the MST, and two trees connected by an edge (u, v) are merged into a single tree.

- On the other hand, if an edge (u, v) connects two vertices in the same tree, then edge (u, v) is discarded.

Source code:

```
import java.utils.Scanner;

public class kruskals {
   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 weighted matrix");
        for(i=1; i<=n; i++)
          for(j=1; j<=n; j++)
            a[i][j]=sc.nextInt();
        kruskals k=new kruskals();
        k.krkl(a ,n);
        sc.close();
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 8 % javac kruskals.java
praveenukkoji@Praveens-MacBook-Pro termwork 8 % java kruskals
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
praveenukkoji@Praveens-MacBook-Pro termwork 8 %
```

B) Prim’s Algorithm

Algorithm:

Choose a node and build a tree from there selecting at every stage the shortest available edge that can extend the tree to an additional node.

- Prim's algorithm has the property that the edges in the set A always form a single tree.
- We begin with some vertex v in a given graph G =(V, E), defining the initial set of vertices A.
- In each iteration, we choose a minimum-weight edge (u, v), connecting a vertex v in the set A to the vertex u outside of set A.

- The vertex u is brought in to A. This process is repeated until a spanning tree is formed.
- Like Kruskal's algorithm, here too, the important fact about MSTs is we always choose the smallest-weight edge joining a vertex inside set A to the one outside the set A.

• The implication of this fact is that it adds only edges that are safe for A; therefore when the algorithm terminates, the edges in set A form a MST

Source code:

```
import java.utils.Scanner;
public class prims {
   public static void main(String[] args) {
      int n=100;
      int size, i, j;
      int u=0,v=0;
      int sum=0;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter Size of Matrix ");
      size=sc.nextInt();
      int a[][] =new int[n][n];
      int visited[] = new int[n];
      System.out.println("Enter the Adjacency matrix ");
      for(i=1; i<=size; i++) {
        for(j=1; j<=size; j++) {
            a[i][j]=sc.nextInt();
        }
      }
      for(i=1; i<=size; i++){
        visited[i]=0;
      }
      System.out.println("Enter the Source ");
      int source=sc.nextInt();
      visited[source]=1;
```

```
int counter=1;
while(counter<=size) {
   int mini=99;
   for(int p=1; p<=size ;p++) {
      for(int q=1; q<=size; q++) {
         if(visited[p]==1 && visited[q]==0) {
            if(p!=q && a[p][q]<mini)
            {
               mini=a[p][q];
               u=p;
               v=q;
            }
         }
      }
   }
   visited[v]=1;
   counter++;
   if(mini!=99) {
      System.out.println(u+"--->"+v+" = "+mini);
      sum = sum + mini;
   }
  }
  System.out.println("Minimum Spanning Tree
  "+sum); sc.close();
 }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 8 % javac prims.java
praveenukkoji@Praveens-MacBook-Pro termwork 8 % java prims
Enter Size of Matrix
6
Enter the Adjacency 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
Enter the Source
1
1--->2 = 3
2--->3 = 1
2--->6 = 4
6--->5 = 2
6--->4 = 5
Minimum Spanning Tree 15
praveenukkoji@Praveens-MacBook-Pro termwork 8 %
```

Experiment No. 9

Write java programs to

A) Implement all pair shortest path problem using **Floyd's** algorithm

Algorithm:

- Accept no .of vertices
- Call graph function to read weighted graph // w(i,j)
- Set D[ ] <- weighted graph matrix // get D {d(i,j)} for k=0
- // If there is a cycle in graph, abort. How to find?
- Repeat for k = 1 to n

Repeat for i = 1 to n
Repeat for j = 1 to n
D[i,j] = min {D[i,j], D[i,k] + D[k,j]} • Print D

Source code:

```
import java.utils.Scanner;

public class floyd {
   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 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();
      floyd f = new floyd();
      f.flyd(a, n);
```

```
        System.out.println("Shortest Path ");
        for(i=1;i<=n;i++)
        {
           for(j=1; j<=n; j++)
              System.out.print(a[i][j] + " ");
           System.out.println();
        }
        sc.close();
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 9 % javac floyd.java
praveenukkoji@Praveens-MacBook-Pro termwork 9 % java floyd
Enter number of vertices
4
Enter the weighted matrix
0 99 3 99
2 0 99 99
99 7 0 1
6 99 99 0
Shortest Path
0 10 3 4
2 0 5 6
7 7 0 1
6 16 9 0
praveenukkoji@Praveens-MacBook-Pro termwork 9 %
```

B) Travelling Sales person problem

Source code:

```
import java.utils.Scanner;

class Travel {
   int weight[][],n, tour[], finalCost;
   final int INF=1000;
   Travel()
   {
Scanner s=new Scanner(System.in);
         System.out.println("Enter no. of nodes:=>");
         n=s.nextInt();
         weight=new int[n][n];
        tour=new int[n-1];
        System.out.println("Enter Adjacency Matrix ");
        for(int i=0;i<n;i++)
           for(int j=0; j<n; j++)
              weight[i][j]=s.nextInt();
        System.out.println();
        System.out.println("Starting node assumed to be node 1.");
        eval();
        s.close();
   }
```

```
public int COST(int curr ,int inputSet[], int setSize)
{
   if(setSize==0)
      return weight[curr][0];
   int min=INF;
   int setNextCOST[]=new int[n-1];
   for(int i=0;i<setSize; i++)
   {
      int k=0;
      for(int j=0; j<setSize;  j++)
      {
         if(inputSet[i]!=inputSet[j])
            setNextCOST[k++]=inputSet[j];
      }
      int temp=COST(inputSet[i],setNextCOST,setSize-1);
      if((weight[curr][inputSet[i]]+temp) < min) {

         min=weight[curr][inputSet[i]]+temp;
      }
   }
   return min;
}

public int MIN(int current, int inputSet[],int setSize)
{
   if(setSize==0)
      return weight[current][0];
```

```
        int min=INF, minindex=0;
        int setNextCOST[]=new int[n-1];
        for(int i=0; i<setSize; i++) //considers each node of inputSet
        {
            int k=0;
            for(int j=0; j<setSize; j++)
            {
                if(inputSet[i]!=inputSet[j])
setNextCOST[k++]=inputSet[j];
            }
            int temp=COST(inputSet[i],setNextCOST,setSize-
            1); if((weight[current][inputSet[i]]+temp) < min) {

                min=weight[current][inputSet[i]]+temp;
                minindex=inputSet[i];
            }
        }
        return minindex;
    }
    public void eval()
    {
        int dummySet[]=new int[n-1];
        for(int i=1;i<n;i++)
            dummySet[i-1]=i;
        finalCost=COST(0,dummySet,n-1);
        constructTour();
    }
```

```
public void constructTour()
{
   int previousSet[]=new int[n-1];
   int nextSet[]=new int[n-2];
   for(int i=1;i<n;i++)
      previousSet[i-1]=i;
   int setSize=n-1;
   tour[0]=MIN(0,previousSet,setSize);
   for(int i=1;i<n-1;i++)
   {
      int k=0;
      for(int j=0; j<setSize; j++)
      {
      if(tour[i-1]!=previousSet[j])
         nextSet[k++]=previousSet[j];
      }
      --setSize;
    tour[i]=MIN(tour[i-1], nextSet, setSize);
      for(int j=0; j<setSize; j++)
      previousSet[j]=nextSet[j];
   }
   display();
}

public void display()
{
   System.out.println();
   System.out.print("The tour is 1-");
```

```
        for(int i=0;i<n-1;i++)
            System.out.print((tour[i]+1)+"-");
        System.out.print("1");
        System.out.println();
        System.out.println("The final cost is "+finalCost);
    }

}
class TravellingSalesman
{
    public static void main(String args[])
    {
        Travel obj =new Travel();
        obj.eval();
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 9 % javac TravellingSalesman.java
praveenukkoji@Praveens-MacBook-Pro termwork 9 % java TravellingSalesman
Enter no. of nodes:=>
4
Enter Adjacency Matrix
0 2 5 7
2 0 8 3
5 8 0 1
7 3 1 0

Starting node assumed to be node 1.

The tour is 1-2-4-3-1
The final cost is 11

The tour is 1-2-4-3-1
The final cost is 11
praveenukkoji@Praveens-MacBook-Pro termwork 9 %
```

Experiment No. 10

Write java programs to

A) Design and implement in java to find a subset of a given set S = {s1, s2, …} 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 solution {1, 2, 6} and {1, 8}. Display a suitable message, if the given problem instance doesn't have a solution.

Source code:

```
import java.utils.Scanner;
import java.lang.Math;

public class Subset {
   static int x[]=new int[100];
   public void subset(int num, int n) {
      for(int i=1;i<=n;i++)
         x[i]=0;
      for(int i=n; num!=0; i--) {
         x[i]=num%2;
         num/=2;
      }
   }

   public static void main(String[] args) {
      int n, d, sum, j, flag=0;
      int a[]=new int[100];
```

```
Scanner sc=new Scanner(System.in);
System.out.println("Enter number of elements
"); n=sc.nextInt();
System.out.println("Enter the elements
"); for(int i=1;i<=n;i++)
   a[i]=sc.nextInt();
System.out.println("Enter val of d ");
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);
      sum=0;
      for(j=1; j<=n; j++) {
           if(x[j]==1)
            sum+=a[j];
      }
      if(d == sum) {
         flag=1;
         System.out.print("Subset is { ");
         for(j=1; j<=n; j++)
         {
            if(x[j]==1)
               System.out.print(a[j]+" ");
         }
         System.out.print("} == "+sum);
         System.out.println();
      }
```

```
        }
        if(flag == 0)
           System.out.println("No Answer");
        sc.close();
        }
    }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 10 % javac Subset.java
praveenukkoji@Praveens-MacBook-Pro termwork 10 % java Subset
Enter number of elements
5
Enter the elements
1 2 5 6 8
Enter val of d
9
Subset is { 1 8 } == 9
Subset is { 1 2 6 } == 9
praveenukkoji@Praveens-MacBook-Pro termwork 10 %
```

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

Source code:

```
import java.utils.*;
class Ham
{
   private int mat[][],x[],n;
   public Ham()
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number of nodes");
      n=sc.nextInt();
      x=new int[n];
      x[0]=0;
      for (int i=1;i<n; i++)
         x[i]=-1;
      mat=new int[n][n];
      System.out.println("Enter the matacency matrix");
      for (int i=0;i<n; i++)
         for (int j=0; j<n; j++)
            mat[i][j]=sc.nextInt();
      sc.close();
   }
```

```
public void nextValue (int v)
{
   int i=0;
   while(true)
   {
      x[v]=x[v]+1;
   if (x[v]==n)
      x[v]=-1;
   if (x[v]==-1)
      return;
   if (mat[x[v-1]][x[v]]==1)
      for (i=0; i<v; i++)
         if (x[i]==x[v])
            break;
   if (i==v)
      if (v<n-1 || v==n-1 && mat[x[n-1]][0]==1)
         return;
   }
}

public void getCycle(int v)
{
   while(true)
   {
      nextValue(v);
      if (x[v]==-1)
         return;
```

```
        if (v==n-1)
        {
            System.out.println("\n Solution : ");
            for (int i=0; i<n; i++)
                System.out.print((x[i]+1)+" ");
            System.out.println(1);
        }
        else getCycle(v+1);
    }
  }
}
class Hamilton
{
  public static void main(String args[])
  {
    Ham obj=new Ham();
    obj.getCycle(1);
  }
}
```

Output:

```
praveenukkoji@Praveens-MacBook-Pro termwork 10 % javac Hamilton.java
praveenukkoji@Praveens-MacBook-Pro termwork 10 % java Hamilton
Enter the number of nodes
6
Enter the matacency matrix
0 1 1 1 0 0
1 0 1 0 0 1
1 1 0 1 1 0
1 0 1 0 1 0
0 0 1 1 0 1
0 1 0 0 1 0

Solution :
1 2 6 5 3 4 1

Solution :
1 2 6 5 4 3 1

Solution :
1 3 2 6 5 4 1

Solution :
1 3 4 5 6 2 1

Solution :
1 4 3 5 6 2 1

Solution :
1 4 5 6 2 3 1
praveenukkoji@Praveens-MacBook-Pro termwork 10 %
```