**Object Oriented Programming**

Question 1

To calculate the distance between 2 point is given by the formula



Where x1,y1 and x2,y2 are the co-ordinates of 2 points.

Write a program to calculate the distance between 2 points

Class name : Distance

Data member : double x,y

Distance() : To initialize the object

void Accept() : To accept values for x and y

double calDistance(Distance x): To calculate the distance between current object and object x and return the distance.

void Display(Distance x1): To print the distance between the current **object and the object X1**

import java.io.\*;

public class Distance {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

double x,y;

Distance()

{

x=0;

y=0;

}

void Accept()throws IOException

{

System.out.println("Enter value for x");

x=Double.parseDouble(br.readLine());

System.out.println("Enter value for y");

y=Double.parseDouble(br.readLine());

}

double calDistance(Distance X)

{

double d;

d=Math.sqrt(Math.pow(x-X.x,2)+Math.pow(y-X.y,2));

return d;

}

void Display(Distance X1)

{

double d=calDistance(X1);

System.out.println("Distance is "+d);

}

public static void main(String args[])throws IOException

{

Distance D1=new Distance();

Distance D2=new Distance();

D1.Accept();

D2.Accept();

D1.Display(D2);

}

}

/\*OUTPUT

Enter value for x

50

Enter value for y

20

Enter value for x

30

Enter value for y

10

Distance is 22.360679774997898

\*/

**Question 2**

**Class name**: AgeCompare

Data members : String n; double a;

AgeCompare():Default constructor

Void Accept():Method to accept name and age.

AgeCompare compare(AgeCompare x) :Method to compare the age in the current object with that of object x and return the object with the greater age.

Void Display():Method to display name and age.

import java.io.\*;

public class AgeCompare {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String n;

double a;

AgeCompare() {

n = "";

a = 0.0;

}

void Accept() throws IOException {

System.out.println("Enter name");

n = br.readLine();

System.out.println("Enter age");

a = Double.parseDouble(br.readLine());

}

AgeCompare Compare(AgeCompare X) {

if (a > X.a) {

return this;

} else {

return X;

}

}

void Display() {

System.out.println("Name is " + n);

System.out.println("Greater Age is " + a);

}

public static void main(String args[]) throws IOException {

AgeCompare A1 = new AgeCompare();

AgeCompare A2 = new AgeCompare();

AgeCompare A3 = new AgeCompare();

A1.Accept();

A2.Accept();

A3 = A1.Compare(A2);

A3.Display();

}

}

**Question 3**

Example program on Call by Value and Call by reference

public class CallByVal {

public static void main(String args[])

{

int x =3;

System.out.println ( "Value of x before calling increment() is "+x);

increment(x);

System.out.println ( "Value of x after calling increment() is "+x);

}

public static void increment ( int a )

{

System.out.println ( "Value of a before incrementing is "+a);

a = a+1;

System.out.println ( "Value of a after incrementing is "+a);

}

}

/\*OUTPUT

Value of x before calling increment() is 3

Value of a before incrementing is 3

Value of a after incrementing is 4

Value of x after calling increment() is 3

\*/

class Number {

int x;

}

class CallByReference {

public static void main ( String[] args ) {

Number a = new Number();

a.x=3;

System.out.println("Value of a.x before calling increment() is "+a.x);

increment(a);

System.out.println("Value of a.x after calling increment() is "+a.x);

}

public static void increment(Number n) {

System.out.println("Value of n.x before incrementing x is "+n.x);

n.x=n.x+1;

System.out.println("Value of n.x after incrementing x is "+n.x);

}

}

/\*OUTPUT

Value of a.x before calling increment() is 3

Value of n.x before incrementing x is 3

Value of n.x after incrementing x is 4

Value of a.x after calling increment() is 4

\*/

**Nested Loops**

a) \*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \*

\* \* \*

\*

import java.io.\*;

public class Diamond {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int i,k,l,n,s;

void Accept()throws IOException

{

System.out.println("Enter a number.");

n=Integer.parseInt(br.readLine());

}

void Display()

{

s=39;

for(k=1;k<=n;k+=2)

{

for(l=1;l<=s;l++)

System.out.print(" ");

for(i=1;i<=k;i++)

{

System.out.print("\*");

}

System.out.println("");

s=s-1;

}

s=s+2;

for(k=n-2;k>=1;k-=2)

{

for(l=1;l<=s;l++)

System.out.print(" ");

for(i=1;i<=k;i++)

{

System.out.print("\*");

}

System.out.println("");

s=s+1;

}

}

public static void main(String args[])throws IOException

{

Diamond D=new Diamond();

D.Accept();

D.Display();

}

}

/\*OUTPUT

Enter a number.

7

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

\*/

----------------------------------------------

b) \* \* \* \* \* \* \*

\* \* \* \* \*

\* \* \*

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

import java.io.\*;

public class Hourglass {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int i,k,l,s,n;

void Accept()throws IOException

{

System.out.println("Enter a number.");

n=Integer.parseInt(br.readLine());

}

void Display()

{

s=39;

for(k=n;k>=1;k-=2)

{

for(l=1;l<=s;l++)

System.out.print(" ");

for(i=1;i<=k;i++)

{

System.out.print("\*");

}

System.out.println("");

s=s+1;

}

s=s-2;

for(k=3;k<=n;k+=2)

{

for(l=1;l<=s;l++)

System.out.print(" ");

for(i=1;i<=k;i++)

{

System.out.print("\*");

}

System.out.println("");

s=s-1;

}

}

public static void main(String args[])throws IOException {

Hourglass H=new Hourglass();

H.Accept();

H.Display();// TODO code application logic here

}

}

/\*OUTPUT

Enter a number.

7

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*/

**Single Dimension Arrays**

1. Union of 2 matrix (AUB)

import java.io.\*;

public class Union {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[10];

int b[]=new int[10];

int c[]=new int[10];

int n,m;

void Accept()throws IOException

{

System.out.println("Enter dimension for A");

n=Integer.parseInt(br.readLine());

System.out.println("Enter dimension for B");

m=Integer.parseInt(br.readLine());

int i;

for(i=0;i<n;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

for(i=0;i<m;i++)

{

System.out.println("Enter a number");

b[i]=Integer.parseInt(br.readLine());

}

for(i=0;i<n;i++)

{

c[i]=a[i];

}

for(i=0;i<m;i++)

{

c[n+i]=b[i];

}

}

void Display()

{

int i;

System.out.println("Array A is:");

for(i=0;i<n;i++)

{

System.out.print(a[i]+" ");

}

System.out.println("");

System.out.println("Array B is:");

for(i=0;i<m;i++)

{

System.out.print(b[i]+" ");

}

System.out.println("");

System.out.println("Union of A and B is:");

for(i=0;i<(n+m);i++)

{

System.out.print(c[i]+" ");

}

}

public static void main(String args[])throws IOException

{

Union U=new Union();

U.Accept();

U.Display();

}

}

/\*OUTPUT

Enter dimension for A

5

Enter dimension for B

3

Enter a number

1

Enter a number

2

Enter a number

3

Enter a number

4

Enter a number

5

Enter a number

6

Enter a number

7

Enter a number

8

Array A is:

1 2 3 4 5

Array B is:

6 7 8

Union of A and B is:

1 2 3 4 5 6 7 8

\*/

1. Intersection of 2 matrix (A Intersection B)

import java.io.\*;

public class InsertionSort {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[5];

void Accept()throws IOException

{

int i;

for(i=0;i<5;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

}

void Sort()

{

int i, j, value;

for(i=1; i<5; i++)

{

value=a[i];

j=i-1;

while(j>=0&&a[j]>value)

{

a[j+1]=a[j];

j--;

}

a[j+1]=value;

}

}

void Display()

{

int i;

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

for(i=0;i<5;i++)

{

System.out.print(a[i]+" ");

}

}

public static void main(String args[])throws IOException

{

InsertionSort I=new InsertionSort();

I.Accept();

I.Sort();

I.Display();

}

}

/\*OUTPUT

Enter a number

89

Enter a number

34

Enter a number

25

Enter a number

36

Enter a number

47

Sorted Array is:

25 34 36 47 89

\*/

1. Selection Sort

import java.io.\*;

public class SelectionSort {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[5];

void Accept()throws IOException

{

int i;

for(i=0;i<5;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

}

void Sort()

{

int sn, ss, t, i, k;

for(i=0; i<=3; i++)

{

sn=a[i]; ss=i;

for(k=i+1; k<=4; k++)

{

if(a[k]<sn)

{

sn=a[k];

ss=k;

}

}

t=a[i];

a[i]=a[ss];

a[ss]=t;

}

}

void Display()

{

int i;

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

for(i=0;i<5;i++)

{

System.out.print(a[i]+" ");

}

}

public static void main(String args[])throws IOException

{

SelectionSort S=new SelectionSort();

S.Accept();

S.Sort();

S.Display();

}

}

/\*OUTPUT

Enter a number

10

Enter a number

40

Enter a number

30

Enter a number

50

Enter a number

20

Sorted Array is:

10 20 30 40 50

\*/

1. Bubble Sort

import java.io.\*;

public class BubbleSort {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[5];

void Accept()throws IOException

{

int i;

for(i=0;i<5;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

}

void Sort()

{

int t, i, k;

for(i=0; i<=3; i++)

{

for(k=0; k<=3-i; k++)

{

if(a[k]>a[k+1])

{

t=a[k];

a[k]=a[k+1];

a[k+1]=t;

}

}

}

}

void Display()

{

int i;

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

for(i=0;i<5;i++)

{

System.out.print(a[i]+" ");

}

}

public static void main(String args[])throws IOException

{

BubbleSort B=new BubbleSort();

B.Accept();

B.Sort();

B.Display();

}

}

/\*OUTPUT

Enter a number

28

Enter a number

90

Enter a number

39

Enter a number

40

Enter a number

72

Sorted Array is:

28 39 40 72 90

\*/

1. Insertion Sort

import java.io.\*;

public class InsertionSort {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[5];

void Accept()throws IOException

{

int i;

for(i=0;i<5;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

}

void Sort()

{

int i, j, value;

for(i=1; i<5; i++)

{

value=a[i];

j=i-1;

while(j>=0&&a[j]>value)

{

a[j+1]=a[j];

j--;

}

a[j+1]=value;

}

}

void Display()

{

int i;

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

for(i=0;i<5;i++)

{

System.out.print(a[i]+" ");

}

}

public static void main(String args[])throws IOException

{

InsertionSort I=new InsertionSort();

I.Accept();

I.Sort();

I.Display();

}

}

/\*OUTPUT

Enter a number

89

Enter a number

34

Enter a number

25

Enter a number

36

Enter a number

47

Sorted Array is:

25 34 36 47 89

\*/

1. Linear Search

import java.io.\*;

public class LSearch {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[5];

int n;

void Accept()throws IOException

{

int i;

for(i=0;i<5;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

System.out.println("Enter the number you are looking for.");

n=Integer.parseInt(br.readLine());

}

void Search()

{

int i;

boolean f=false;

for(i=0;i<5;i++)

{

if(n==a[i])

{

f=true;

System.out.println("It is found in subscript "+i);

}

}

if(f==false)

System.out.println("No such number in array.");

}

public static void main(String args[])throws IOException

{

LSearch L=new LSearch();

L.Accept();

L.Search();

}

}

/\*OUTPUT

Enter a number

1

Enter a number

2

Enter a number

3

Enter a number

4

Enter a number

5

Enter the number you are looking for.

2

It is found in subscript 1

\*/

1. Binary Search

import java.io.\*;

public class BSearch {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[5];

int n;

void Accept()throws IOException

{

int i;

for(i=0;i<5;i++)

{

System.out.println("Enter a number");

a[i]=Integer.parseInt(br.readLine());

}

System.out.println("Enter the number you are looking for.");

n=Integer.parseInt(br.readLine());

}

void Search()

{

int low,mid,high;

low=0;

high=5;

boolean f=false;

while(low<=high)

{

mid=(low+high)/2;

if(n==a[mid])

{

System.out.println("It is found in subscript "+mid);

f=true;

break;

}

else if(n<a[mid])

high=mid-1;

else if(n>a[mid])

low=mid+1;

}

if(f==false)

System.out.println("No such number in array.");

}

public static void main(String args[])throws IOException

{

BSearch B=new BSearch();

B.Accept();

B.Search();

}

}

/\*OUTPUT

Enter a number

23

Enter a number

34

Enter a number

45

Enter a number

56

Enter a number

67

Enter the number you are looking for.

12

No such number in array.

\*/

1. Menu driven to perform deletion and insertion of elements from the arrays

import java.io.\*;

public class ArrMenu {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[]=new int[10];

int c=0;

void Insert(int num)

{

a[c++]=num;

}

void Delete(int p)

{

int i;

for(i=p; i<9; i++)

{

a[i]=a[i+1];

}

}

void Display()

{

int i;

for(i=0;i<10;i++)

{

if(a[i]!=0)

System.out.print(a[i]+" ");

}

System.out.println("");

}

void Menu()throws IOException

{

int ch;

while(true)

{

System.out.println("1. Insert");

System.out.println("2. Delete");

System.out.println("3. Display");

System.out.println("4. Exit");

System.out.println("Enter your choice");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter number");

int num = Integer.parseInt(br.readLine());

Insert(num);

break;

case 2:

System.out.println("Enter position of number");

int p = Integer.parseInt(br.readLine());

Delete(p);

break;

case 3:

Display();

break;

case 4:

System.exit(0);

default:

System.out.println("Wrong Entry");

}

}

}

public static void main(String args[])throws IOException

{

ArrMenu A=new ArrMenu();

A.Menu();

}

}

/\*OUTPUT

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

1

Enter number

10

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

1

Enter number

20

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

1

Enter number

30

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

3

10 20 30

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

2

Enter position of number

1

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

3

10 30

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice

4

\*/

1. Implementation of Stack using arrays

import java.io.\*;

public class ArrStack {

int a[]=new int[5];

int top=-1;

void push(int num)

{

if (top < 4)

a[++top] = num;

else

{

System.out.println("Stack Full");

}

}

void pop()

{

if (top >= 0)

{

System.out.println("Popped value: "+a[top]);

top--;

}

else

System.out.println("Stack Empty");

}

void Display()

{

if (top < 0)

System.out.println("No elements in stack");

else

{

System.out.println("Elements in stack :");

for (int i = 0; i <= top; i++)

{

System.out.print(a[i]+" ");

}

System.out.println("");

}

}

public static void main(String[] args)throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

ArrStack S = new ArrStack();

int ch;

while(true)

{

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");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter number");

int num=Integer.parseInt(br.readLine());

S.push(num);

break;

case 2:

S.pop();

break;

case 3:

S.Display();

break;

case 4:

System.exit(0);

break;

default:

System.out.println("Wrong entry");

}

}

}

}

/\*OUTPUT

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

11

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

22

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 22

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

33

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

44

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

55

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

66

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

77

Stack Full

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

3

Elements in stack :

11 33 44 55 66

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 66

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 55

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 44

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 33

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 11

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Stack Empty

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

3

No elements in stack

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

4

\*/

1. Implementation of Queue using arrays

import java.io.\*;

public class ArrQueue

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int a[]=new int[5];

int r,f; //r=rear and f=front

final int max=5;

ArrQueue()

{

f=0;r=-1;//r=-1

}

void Insert(int n)

{

if(r==(max-1))

System.out.println("Queue Full");

else

a[++r]=n;

}

void Delete()

{

int n=0;

if(f>r)

System.out.println("Queue Empty");

else

{

n=a[f++];

System.out.println("Element deleted is "+n);

}

}

void Display()

{

if(f>r)

System.out.println("Queue Empty");

else

{

for (int i=f;i<=r;i++)

System.out.print(a[i]+" ");

System.out.println();

}

}

void Menu()throws IOException

{

int n,num;

while(true)

{

System.out.println("1.Insert");

System.out.println("2.Delete");

System.out.println("3.Display");

System.out.println("4.Exit");

System.out.println("Enter your choice");

n=Integer.parseInt(br.readLine());

switch(n)

{

case 1:

System.out.println("Enter the number you wish to store on queue" );

num=Integer.parseInt(br.readLine());

Insert(num);

break;

case 2:

Delete();

break;

case 3:

Display();

break;

case 4:

System.exit(0);

default:

System.out.println("Wrong entry");

}

}

}

public static void main(String args[])throws IOException

{

ArrQueue Q=new ArrQueue();

Q.Menu();

}

}

/\*OUTPUT

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

10

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

20

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

30

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

40

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

50

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

60

Queue Full

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

3

10 20 30 40 50

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Element deleted is 10

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Element deleted is 20

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Element deleted is 30

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Element deleted is 40

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Element deleted is 50

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Queue Empty

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

3

Queue Empty

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

4

\*/

1. Implementation of Dqueue using arrays

import java.io.\*;

public class ArrDQ

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int a[]=new int[5];

int r,f;

final int max=5;

ArrDQ()

{

f=0;r=-1;

}

void InsertF(int n)

{

if(r==0)

System.out.println("Front Insertion not possible.");

else if(f!=0)

a[--f]=n;

else

a[++r]=n;

}

void InsertE(int n)

{

if(r==(max-1))

System.out.println("Queue Full");

else

a[++r]=n;

}

void DeleteF()

{

int n;

if(f>r)

System.out.println("Q Empty");

else

{

n=a[f++];

System.out.println("Number deleted from the beginning "+n);

}

}

void DeleteE()

{

int n;

if(f>r)

System.out.println("Q Empty");

else

{

n=a[r--];

System.out.println("Number deleted from the end "+n);

}

}

void Display()

{

if(f>r)

System.out.println("Queue Empty");

else

{

for (int i=f;i<=r;i++)

System.out.print(a[i]+" ");

System.out.println();

}

}

void Menu()throws IOException

{

int n,num;

while(true)

{

System.out.println("1.InsertF");

System.out.println("2.InsertE");

System.out.println("3.DeleteF");

System.out.println("4.DeleteE");

System.out.println("5.Display");

System.out.println("6.Exit");

System.out.println("Enter your choice");

n=Integer.parseInt(br.readLine());

switch(n)

{

case 1:

System.out.println("Enter the number you wish to store at the beginning");

num=Integer.parseInt(br.readLine());

InsertF(num);

break;

case 2:

System.out.println("Enter the number you wish to store at the end" );

num=Integer.parseInt(br.readLine());

InsertE(num);

break;

case 3:

DeleteF();

break;

case 4:

DeleteE();

break;

case 5:

Display();

break;

case 6:

System.exit(0);

default:

System.out.println("Wrong entry");

}

}

}

public static void main(String args[])throws IOException

{

ArrDQ Q=new ArrDQ();

Q.Menu();

}

}

/\*OUTPUT

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

1

Enter the number you wish to store at the beginning

10

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

1

Enter the number you wish to store at the beginning

20

Front Insertion not possible.

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

2

Enter the number you wish to store at the end

30

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

1

Enter the number you wish to store at the beginning

40

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

5

10 30 40

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

3

Number deleted from the beginning 10

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

4

Number deleted from the end 40

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

3

Number deleted from the beginning 30

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

4

Q Empty

1.InsertF

2.InsertE

3.DeleteF

4.DeleteE

5.Display

6.Exit

Enter your choice

6

\*/

**Double Dimension Arrays**

1. Print the transpose of Matrix

import java.io.\*;

public class MTranspose {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[][]=new int[5][3];

void Accept()throws IOException

{

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<3;k++)

{

System.out.println("Enter a number");

a[i][k]=Integer.parseInt(br.readLine());

}

}

System.out.println("Normal Matrix is:");

for(i=0;i<5;i++)

{

for(k=0;k<3;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

}

void Trans()

{

int i,k;

System.out.println("Transposed Matrix is:");

for(i=0;i<3;i++)

{

for(k=0;k<5;k++)

{

System.out.print(a[k][i]+" ");

}

System.out.println("");

}

}

public static void main(String args[])throws IOException

{

MTranspose M=new MTranspose();

M.Accept();

M.Trans();

}

}

/\*OUTPUT

Enter a number

01

Enter a number

02

Enter a number

03

Enter a number

04

Enter a number

05

Enter a number

06

Enter a number

07

Enter a number

08

Enter a number

09

Enter a number

10

Enter a number

11

Enter a number

12

Enter a number

13

Enter a number

14

Enter a number

15

Normal Matrix is:

1 2 3

4 5 6

7 8 9

10 11 12

13 14 15

Transposed Matrix is:

1 4 7 10 13

2 5 8 11 14

3 6 9 12 15

\*/

1. Matrix Addition

import java.io.\*;

public class MAdd {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[][]=new int[2][3];

void Accept()throws IOException

{

int i,k;

for(i=0;i<2;i++)

{

for(k=0;k<3;k++)

{

System.out.println("Enter a number");

a[i][k]=Integer.parseInt(br.readLine());

}

}

System.out.println("Matrix is:");

for(i=0;i<2;i++)

{

for(k=0;k<3;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

}

MAdd Add(MAdd X)

{

MAdd Z=new MAdd();

int i,k;

for(i=0;i<2;i++)

{

for(k=0;k<3;k++)

{

Z.a[i][k]=a[i][k]+X.a[i][k];

}

}

return Z;

}

void Display()

{

int i,k;

System.out.println("Addition of Matrix:");

for(i=0;i<2;i++)

{

for(k=0;k<3;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

}

public static void main(String args[])throws IOException

{

MAdd M1=new MAdd();

MAdd M2=new MAdd();

MAdd M3=new MAdd();

M1.Accept();

M2.Accept();

M3=M1.Add(M2);

M3.Display();

}

}

/\*OUTPUT

Enter a number

1

Enter a number

2

Enter a number

3

Enter a number

4

Enter a number

5

Enter a number

6

Matrix is:

1 2 3

4 5 6

Enter a number

10

Enter a number

20

Enter a number

30

Enter a number

40

Enter a number

50

Enter a number

60

Matrix is:

10 20 30

40 50 60

Addition of Matrix:

11 22 33

44 55 66

\*/

1. Matrix Multiplication

import java.io.\*;

public class MMul {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[][]=new int[2][2];

void Accept()throws IOException

{

int i,k;

for(i=0;i<2;i++)

{

for(k=0;k<2;k++)

{

System.out.println("Enter a number");

a[i][k]=Integer.parseInt(br.readLine());

}

}

System.out.println("Matrix is:");

for(i=0;i<2;i++)

{

for(k=0;k<2;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

}

MMul Mul(MMul X)

{

MMul Z=new MMul();

int i,k;

for(i=0;i<2;i++)

{

for(k=0;k<2;k++)

{

Z.a[i][k]=(a[i][0]\*X.a[0][k])+(a[i][1]\*X.a[1][k]);

}

}

return Z;

}

void Display()

{

int i,k;

System.out.println("Multiplication of Matrix:");

for(i=0;i<2;i++)

{

for(k=0;k<2;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

}

public static void main(String args[])throws IOException

{

MMul M1=new MMul();

MMul M2=new MMul();

MMul M3=new MMul();

M1.Accept();

M2.Accept();

M3=M1.Mul(M2);

M3.Display();

}

}

/\*OUTPUT

Enter a number

1

Enter a number

2

Enter a number

3

Enter a number

4

Matrix is:

1 2

3 4

Enter a number

5

Enter a number

6

Enter a number

7

Enter a number

8

Matrix is:

5 6

7 8

Multiplication of Matrix:

19 22

43 50

\*/

**Question 4 (2003)**

A wondrous square is an n by n grid which fulfils the following conditions:

* It contains integers from 1 to n2, where each integer appear only once
* The sum of integer in any row or column must add up to 0.5x n x (n2 + 1)

Write a program to read n (2 <= n <=10) and the values stored in these n by n cells and output if the grid represents a wondrous square or not

Also output all the prime numbers in the grid along with their row index and column index as shown in the output.

**Sample Input**

N = 4

16 15 1 2

6 4 10 14

9 8 12 5

3 7 11 13

**Output:**

YES IT REPRESENTS A WONDROUS SQUARE

PRIME ROW INDEX COLUMN INDEX

2 0 3

3 3 0

5 2 3

7 3 1

11 3 2

13 3 3

import java.io.\*;

public class WondrousSq {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int n;

int a[][]=new int[10][10];

void Accept()throws IOException

{

System.out.println("Enter a value for N");

n=Integer.parseInt(br.readLine());

int i,k;

for(i=0;i<n;i++)

{

for(k=0;k<n;k++)

{

System.out.println("Enter a number");

a[i][k]=Integer.parseInt(br.readLine());

}

}

}

void CheckWonder()

{

boolean f=true;

int i,k,c,d;

double sumr, sumc;

sumr=sumc=0.0;

double value=0.5\*n\*(Math.pow(n,2)+1);

for(i=0;i<n;i++)

{

for(k=0;k<n;k++)

{

for(c=i+1;c<n;c++)

{

for(d=k+1;d<n;d++)

{

if(a[i][k]==a[c][d]) //integer is repeated

f=false;

}

}

}

}

for(i=0;i<n;i++)

{

for(k=0;k<n;k++)

{

sumr=sumr+a[i][k];

sumc=sumc+a[k][i];

}

if(sumr!=value||sumc!=value) //the sum of row or column does not satisfy wondrous condition

f=false;

sumr=sumc=0;

}

if(f==true)

System.out.println("YES IT REPRESENTS A WONDROUS SQUARE");

else

System.out.println("NO IT DOES NOT REPRESENT A WONDROUS SQUARE");

}

boolean CheckPrime(int x)

{

int i;

int div=2;

for(i=2;i<x;i++)

{

if(x%i==0)

div++;

}

if(div==2&&x!=1) //1 is not a prime number

return true;

else

return false;

}

void PrintPrime()

{

int i,k;

System.out.println("PRIME ROW INDEX COLUMN INDEX");

for(i=0;i<n;i++)

{

for(k=0;k<n;k++)

{

if(CheckPrime(a[i][k]))

{

if(a[i][k]<=9)

System.out.println(" "+a[i][k]+" "+i+" "+k);

else

System.out.println(" "+a[i][k]+" "+i+" "+k);

}

}

}

}

public static void main(String args[])throws IOException

{

WondrousSq W=new WondrousSq();

W.Accept();

W.CheckWonder();

W.PrintPrime();

}

}

/\*OUTPUT

Enter a value for N

4

Enter a number

16

Enter a number

15

Enter a number

1

Enter a number

2

Enter a number

6

Enter a number

4

Enter a number

10

Enter a number

14

Enter a number

9

Enter a number

8

Enter a number

12

Enter a number

5

Enter a number

3

Enter a number

7

Enter a number

11

Enter a number

13

YES IT REPRESENTS A WONDROUS SQUARE

PRIME ROW INDEX COLUMN INDEX

2 0 3

5 2 3

3 3 0

7 3 1

11 3 2

13 3 3

\*/

**Question 5**

Class name : DDA1

Data member : int a[][]=new int[5][5]

DDA1() : Constructor to initialize data members

void Accept() : To accept 5x5 numbers in the array

void Display1() : Display all the elements above the left diagonal

void Display2() : Display all the elements below the left diagonal

void Display3() : Display all the elements above the right diagonal

void Display4() : Display all the elements above the right diagonal

void Swap1() : Swaps the elements above the left diagonal with the elements below the left diagonal

void Swap2() : Swaps the elements above the right diagonal with the elements below the right diagonal

import java.io.\*;

public class DDA1 {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a[][]=new int[5][5];

DDA1()

{

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

a[i][k]=0;

}

}

}

void Accept()throws IOException

{

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

System.out.println("Enter a number");

a[i][k]=Integer.parseInt(br.readLine());

}

}

}

void Display1()

{

System.out.println("DISPLAY 1");

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

if(k>i)

System.out.print(a[i][k]+" ");

else

System.out.print(" ");

}

System.out.println("");

}

System.out.println("");

}

void Display2()

{

System.out.println("DISPLAY 2");

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

if(k<i)

System.out.print(a[i][k]+" ");

else

System.out.print(" ");

}

System.out.println("");

}

System.out.println("");

}

void Display3()

{

System.out.println("DISPLAY 3");

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

if(i+k<4)

System.out.print(a[i][k]+" ");

else

System.out.print(" ");

}

System.out.println("");

}

System.out.println("");

}

void Display4()

{

System.out.println("DISPLAY 4");

int i,k;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

if(i+k>4)

System.out.print(a[i][k]+" ");

else

System.out.print(" ");

}

System.out.println("");

}

System.out.println("");

}

void Swap1()

{

System.out.println("SWAP 1");

int i,k;

int t;

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

if(k>i)

{

t=a[i][k];

a[i][k]=a[k][i];

a[k][i]=t;

}

}

}

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

System.out.println("");

}

void Swap2() //if called after swap 1, this method swaps the new matrix obtained after swap 1

{

System.out.println("SWAP 2");

int i,k;

int t;

int count;

for(i=0;i<5;i++)

{

count=4-i;

for(k=0;k<5;k++)

{

if(i+k<4)

{

t=a[i][k];

a[i][k]=a[i+count][k+count];

a[i+count][k+count]=t;

count=count-1;

}

}

}

for(i=0;i<5;i++)

{

for(k=0;k<5;k++)

{

System.out.print(a[i][k]+" ");

}

System.out.println("");

}

System.out.println("");

}

public static void main(String args[])throws IOException

{

DDA1 D=new DDA1();

D.Accept();

D.Display1();

D.Display2();

D.Display3();

D.Display4();

D.Swap1();

D.Swap2();

}

}

/\*OUTPUT

Enter a number

10

Enter a number

20

Enter a number

30

Enter a number

40

Enter a number

50

Enter a number

60

Enter a number

70

Enter a number

80

Enter a number

90

Enter a number

10

Enter a number

11

Enter a number

12

Enter a number

13

Enter a number

14

Enter a number

15

Enter a number

16

Enter a number

17

Enter a number

18

Enter a number

19

Enter a number

20

Enter a number

21

Enter a number

22

Enter a number

23

Enter a number

24

Enter a number

25

DISPLAY 1

20 30 40 50

80 90 10

14 15

20

DISPLAY 2

60

11 12

16 17 18

21 22 23 24

DISPLAY 3

10 20 30 40

60 70 80

11 12

16

DISPLAY 4

10

14 15

18 19 20

22 23 24 25

SWAP 1

10 60 11 16 21

20 70 12 17 22

30 80 13 18 23

40 90 14 19 24

50 10 15 20 25

SWAP 2

25 24 23 22 21

20 19 18 17 16

15 14 13 12 11

10 90 80 70 60

50 40 30 20 10

\*/

**Modulus**

**Question 1**

**Class name** : AmicableNumbers

**Data member** : int num

**void Input(int n1)** : Parameterized method to initialize num = n1

**int SumFactors(int n1):** To find the sum of all the factors of n1 other than n1 and return it

**boolean checkAmicable(AmicableNumbers x)** : Checks if the number in the current object and the number in object x are Amicable to each other then it returns true else it returns false

**void Display(AmicableNumbers x):** Displays if the number in the current object and the number in the object x are Amicable to each other or not.

import java.io.\*;

public class AmicableNumbers {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int num;

void Input(int n1)

{

num=n1;

}

int SumFactors(int n1)

{

int i;

int sum=0;

for(i=1;i<n1;i++)

{

if(n1%i==0)

sum=sum+i;

}

return sum;

}

boolean checkAmicable(AmicableNumbers x)

{

int sum1, sum2;

sum1=SumFactors(num);

sum2=SumFactors(x.num);

if(num==sum2&&x.num==sum1)

return true;

else

return false;

}

void Display(AmicableNumbers x)

{

if(checkAmicable(x))

System.out.println(num+" and "+x.num+" are Amicable Numbers.");

else

System.out.println(num+" and "+x.num+" are not Amicable Numbers.");

}

public static void main(String args[])throws IOException

{

AmicableNumbers A1=new AmicableNumbers();

AmicableNumbers A2=new AmicableNumbers();

A1.Input(220);

A2.Input(284);

A1.Display(A2);

}

}

/\*OUTPUT

220 and 284 are Amicable Numbers.

\*/

**Question 2(2007)**

A unique-digit integer is a positive number (without leading zeros)with no duplicate digits. For eg. 7, 135, 214 are all unique-digit integers whereas 33, 3121, 300 are not

Given two positive integer m and n, where m<n. write a program to determine how many unique- digits are there in the range between m and n(both inclusive) and output them

The input contains two positive integers m and n. Assume n<30000 and n<30000. you are to output the number of unique-digit integers integers in the specified range along with their values in the format specified below:

Sample Data:

**INPUT**

m=100

n=120

**Output:**

THE UNIQUE\_DIGIT INTEGERS ARE:-

102, 103, 104, 105, 106,107, 108, 109, 120

Frequency of unique digits integer is : 9

**INPUT**

M=2500

N=2550

THE UNIQUE\_DIGIT INTEGERS ARE:-

2501, 2503, 2504, 2506, 2507, 2508, 2509, 2510, 2513, 2514,2516, 2517, 2518, 2519, 2530, 2531,2534, 2536, 2537, 2538, 2539, 2540, 2541, 2543, 2546, 2547, 2548, 2549

Frequency of unique digits integer is : 28

import java.io.\*;

public class UniqueDig {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int m,n;

void Accept()throws IOException

{

System.out.println("Enter a positive integer value for M which is less than 3000.");

m=Integer.parseInt(br.readLine());

System.out.println("Enter a positive integer value for N which is less than 3000 and greater than m.");

n=Integer.parseInt(br.readLine());

}

boolean checkUnique(int x)

{

boolean f=true;

if(x%1000==0) //leading zeros

f=false;

int r;

int count[]={0,0,0,0,0,0,0,0,0,0};

while(x!=0)

{

r=x%10;

count[r]++;

if(count[r]>1) //digit repeated

{

f=false;

break;

}

x=x/10;

}

return f;

}

void Control()

{

System.out.println("THE UNIQUE\_DIGIT INTEGERS ARE:-");

int i;

int c=0;

String s="";

for(i=m;i<=n;i++)

{

if(checkUnique(i))

{

s=s+i+", ";

c++;

}

}

int l=s.length();

s=s.substring(0,(l-2)); // to remove the last comma and space

System.out.println(s);

System.out.println("Frequency of unique digit intergers is: "+c);

}

public static void main(String args[])throws IOException

{

UniqueDig U=new UniqueDig();

U.Accept();

U.Control();

}

}

/\*OUTPUT

Enter a positive integer value for M which is less than 3000.

2500

Enter a positive integer value for N which is less than 3000 and greater than m.

2550

THE UNIQUE\_DIGIT INTEGERS ARE:-

2501, 2503, 2504, 2506, 2507, 2508, 2509, 2510, 2513, 2514, 2516, 2517, 2518, 2519, 2530, 2531, 2534, 2536, 2537, 2538, 2539, 2540, 2541, 2543, 2546, 2547, 2548, 2549

Frequency of unique digit intergers is: 28

\*/

**Question 3 (Smith number)**

A smith number is a composite number, the sum of whose digits is the sum of the digits of its prime factors obtained as a result of prime factorization (excluding 1). The first few such numbers are 4, 22, 27, 58, 85, 94, 121,….

**Example:**

1. **666**

Prime factors are 2, 3, 3 and 37

Sum of the digits are (6+6+6) = 18

Sum of the digits of the factors (2+3+3+(3+7))=18

1. 4937775

Prime factors are 3,5,5, 65837

Sum of the digits are (4+9+3+7+7+7+5)=42

Sum of the digits of the factors (3+5+5+(6+5+8+3+7))=42

Write a program to input a number and display whether the number is a Smith number or not.

**Sample Data :**

Input 94 Output SMITH Number

Input 102 Output NOT SMITH Number

Input 666 Output SMITH Number

Input 999 Output NOT SMITH Number

import java.io.\*;

public class SmithNum {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int n;

void Accept()throws IOException

{

System.out.println("Enter a number");

n=Integer.parseInt(br.readLine());

}

int sumDig(int x)

{

int r;

int sum=0;

while(x!=0)

{

r=x%10;

sum=sum+r;

x=x/10;

}

return sum;

}

boolean checkPrime(int x)

{

int i;

int c=0;

for(i=1;i<=x;i++)

{

if(x%i==0)

c++;

}

if(c==2)

return true;

else

return false;

}

int PrimeFact(int x)

{

int i=2;

int sumfact=0;

while(x>1)

{

if(x%i==0)

{

if(checkPrime(i))

{

sumfact=sumfact+sumDig(i);

x=x/i;

}

}

else

i++;

}

return sumfact;

}

boolean checkSmith(int x)

{

if(sumDig(x)==PrimeFact(x))

return true;

else

return false;

}

void Display()

{

if(checkSmith(n))

System.out.println("SMITH Number.");

else

System.out.println("NOT SMITH Number.");

}

public static void main(String args[])throws IOException

{

SmithNum S=new SmithNum();

S.Accept();

S.Display();

}

}

/\*OUTPUT

Enter a number

94

sumDig(x)=13

PrimeFact(x)=13

SMITH Number.

\*/

**Question 4(2010)**

A positive whole number ‘n’ that has d number of digits is squared and split into two pieces, a right- had piece that has ‘d’ digits and left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of two pieces is equal to the number, then ‘n’ is a Kaprekar number. The first few Kaprekar numbers are 9, 45, 297,…..

Example 1

9

92 = 81, right-hand piece of 81 = 1 and left hand piece if 81 = 8

Sum = 1 + 8 = 9 i.e equal to the number

Example 2

45

452= 2025

Sum = 20+25= 45, ie equal to the number

Example 3

297

2972=88209

Sum = 209 + 88 = 297, ie equal to the number, right-hand piece of 88209 = 209 and left hand piece if 88209 = 88.

Given 2 positive integers p and q, where p<q, write a program to determine how many Kaprekar numbers are there in the range between p and q(both inclusive) and output them.

The number contains 2 positive integers p and q. Assume p<5000 and q<5000. You are to output the number of Kaprekar numbers in the specified range along with their values in the format specified below:

**SAMPLE DATA:**

**INPUT**

p=1

q=1000

**OUTPUT**

THE KAPREKAR NUMBERS ARE:

1, 9, 45, 55, 99, 297, 703, 999

FREQUENCY OF KAPREKAR NUMBERS IS : 8

import java.io.\*;

public class Kaprekar {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int p,q;

void Accept()throws IOException

{

System.out.println("Enter a positive integer value for P which is less than 5000.");

p=Integer.parseInt(br.readLine());

System.out.println("Enter a positive integer value for Q which is less than 5000 and greater than P.");

q=Integer.parseInt(br.readLine());

}

boolean checkKap(int x)

{

int sq, length, mid, l, r;

String s, left, right;

sq=x\*x;

s=Integer.toString(sq);

if(sq<=9)

s="0"+sq;

length=s.length();

mid=length/2;

left=s.substring(0,mid);

right=s.substring(mid);

l=Integer.parseInt(left);

r=Integer.parseInt(right);

if(l+r==x)

return true;

else

return false;

}

void Control()

{

System.out.println("THE KAPREKAR NUMBERS ARE:");

int i;

int c=0;

String s="";

for(i=p;i<=q;i++)

{

if(checkKap(i))

{

s=s+i+", ";

c++;

}

}

int l=s.length();

s=s.substring(0,(l-2)); // to remove the last comma and space

System.out.println(s);

System.out.println("FREQUENCY OF KAPREKAR NUMBERS IS: "+c);

}

public static void main(String args[])throws IOException

{

Kaprekar K=new Kaprekar();

K.Accept();

K.Control();

}

}

/\*OUTPUT

Enter a positive integer value for P which is less than 5000.

1

Enter a positive integer value for Q which is less than 5000 and greater than P.

1000

THE KAPREKAR NUMBERS ARE:

1, 9, 45, 55, 99, 297, 703, 999

FREQUENCY OF KAPREKAR NUMBERS IS: 8

\*/

**STRING**

**Question 1**

Write a program which takes a sting (maximum 80 characters) terminate by a

full stop. The words in this string are assumed to be separated by one or more blanks.

Arrange the words of the input string in descending order of their lengths.

Same length words should be stored alphabetically. Each word must start with

an uppercase letter and the sentence should be terminated by a full stop.

Test your program for the following data and some random data.

SAMPLE DATA :

INPUT :

“ This is human resource department ”

OUTPUT :

Department Resource Human This is.

INPUT :

“To handle yourself use your head and to handle others use your heart.”

OUTPUT :

Yourself Handle Handle Others Heart Head Your Your And Use Use To To.

import java.io.\*;

import java.util.\*;

public class String1 {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String s;

void Accept()throws IOException

{

System.out.println("Enter a String of max 80 characters");

s=br.readLine();

}

void Arrange()

{

String t, ans;

ans="";

int i,k,l1,l2,l3;

s=s.replace(".", " ");

StringTokenizer st=new StringTokenizer(s);

int c=st.countTokens();

String a[]=new String[c];

for(i=0;i<c;i++)

{

a[i]=st.nextToken();

}

for(i=0;i<=c-2;i++)

{

for(k=i+1;k<c;k++)

{

l1=a[i].length();

l2=a[k].length();

if(l2>l1) //descending order of length

{

t=a[i];

a[i]=a[k];

a[k]=t;

}

else if(l1==l2) //if same length then alphabetical order

{

if(a[k].compareTo(a[i])<0)

{

t=a[i];

a[i]=a[k];

a[k]=t;

}

}

}

}

for(i=0;i<c;i++)

{

a[i] = a[i].substring(0,1).toUpperCase() + a[i].substring(1); //to capitalize first letter of word

ans=ans+a[i]+" ";

}

l3=ans.length();

ans=ans.substring(0,l3-1); //to remove last space

ans=ans+"."; //fullstop at the end of sentence

System.out.println(ans);

}

public static void main(String args[])throws IOException

{

String1 S=new String1();

S.Accept();

S.Arrange();

}

}

/\*OUTPUT

Enter a String of max 80 characters

To handle yourself use your head and to handle others use your heart.

Yourself Handle Handle Others Heart Head Your Your And Use Use To To.

\*/

**Question 2**

The input in this problem will consists of a number of lines of English text

consisting of the letters of the English alphabet, the punctuation marks (‘)

apostrophe, (.) full stop, (,) comma, (;)semicolon, (:) colon and white space

characters (blank, newline). Your task is to print the word of the text in reverse

order without a punctuation marks other than blanks.

For example consider the following input text:

This is a sample piece of text to illustrate this problem. If you are smart you

will solve this right.

The corresponding output would read as:

right this solve will you smart are you If problem this illustrate to text of piece

sample a is This.

that is , the lines are printed in reverse order.

**Note: Individual words are not reversed.**

**Input format**

The first line of input contains a single integer *N ( < = 20 )*, indicating the

number of lines in the input. This is followed by N lines of input text. Each line

should accept a maximum of 80characters.

**Output format**

Output the text containing the input lines in reversed order without punctuation

except blanks illustrated above.

Test your program for the following data and some random data.

**SAMPLE DATA**

**INPUT:**

**2**

Emotions, controlled and directed to work, is character. By Swami Vivekananda.

**OUTPUT:**

Vivekananda Swami By character is work to directed and controlled Emotions.

**INPUT:**

1

Do not judge a book by its cover.

**OUTPUT**

cover its by book a judge not Do.

import java.io.\*;

import java.util.\*;

public class String2 {

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int n;

String s;

void Accept()throws IOException

{

System.out.println("Enter the number of sentences.");

n=Integer.parseInt(br.readLine());

System.out.println("Enter a String.");

s=br.readLine();

}

void ReverseSent()

{

s=s.replace("'"," ");

s=s.replace("."," ");

s=s.replace(","," ");

s=s.replace(";"," ");

s=s.replace(":"," ");

s=s.replace("/n"," ");

StringTokenizer st=new StringTokenizer(s);

int c=st.countTokens();

String a[]=new String[c];

int i;

for(i=0;i<c;i++)

{

a[i]=st.nextToken();

}

for(i=c-1;i>=0;i--)

{

System.out.print(a[i]+" ");

}

}

public static void main(String args[])throws IOException

{

String2 S=new String2();

S.Accept();

S.ReverseSent();

}

}

/\*OUTPUT

Enter the number of sentences.

2

Enter a String.

Emotions, controlled and directed to work, is character. By Swami Vivekananda.

Vivekananda Swami By character is work to directed and controlled Emotions

\*/

**Date**

1. class Name : MyDate

Data member : int dd,mm,yy

MyDate() : Constructor to initialize data member

void Accept() : To accept day, month and year

int julianDate() : To find the number of days between the beginning of the given year to the given date.

void Display() : To print the Julian date

import java.io.\*;

public class MyDate {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int dd, mm, yy;

MyDate()

{

dd=mm=yy=0;

}

void Accept()throws IOException

{

System.out.println("Enter date in dd/mm/yy format.");

String s=br.readLine();

dd=Integer.parseInt(s.substring(0,2));

mm=Integer.parseInt(s.substring(3,5));

yy=Integer.parseInt(s.substring(6));

}

int julianDate()

{

int max[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

if(yy % 4 == 0)

{

max[2] = 29;

}

int JD=0; int i;

for(i=1;i<=mm-1;i++)

{

JD=JD+max[i];

}

JD=JD+dd;

return JD;

}

void Display()

{

System.out.println("The Julian Date is "+julianDate());

}

public static void main(String args[])throws IOException

{

MyDate M=new MyDate();

M.Accept();

M.Display();

}

}

/\*OUTPUT

Enter date in dd/mm/yy format.

18/05/04

The Julian Date is 139

\*/

1. class Name : MyDate2

Data member : int dd,mm,yy

MyDate2() : Constructor to initialize data member

void Accept() : To accept day, month and year

int diff(MyDate2 X) : Difference between current object and object X. Assuming the year in both the dates are **same.**

void Display(MyDate2 X) : To print the difference between 2 dates

import java.io.\*;

public class MyDate2 {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int dd, mm, yy;

String s;

MyDate2()

{

dd=mm=yy=0;

}

void Accept()throws IOException

{

System.out.println("Enter date in dd/mm/yy format.");

s=br.readLine();

dd=Integer.parseInt(s.substring(0,2));

mm=Integer.parseInt(s.substring(3,5));

yy=Integer.parseInt(s.substring(6));

}

int diff(MyDate2 X)

{

int max[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};

if(yy % 4 == 0)

{

max[2] = 29;

}

int diff=0; int i;

for(i=mm;i<=X.mm-1;i++)

{

diff=diff+max[i];

}

diff=diff-dd+X.dd;

return diff;

}

void Display(MyDate2 X)

{

System.out.println("The difference between dates is "+diff(X));

}

public static void main(String args[])throws IOException

{

MyDate2 M1=new MyDate2();

MyDate2 M2=new MyDate2();

M1.Accept();

M2.Accept();

M1.Display(M2);

}

}

/\*OUTPUT

Enter date in dd/mm/yy format.

18/02/16

Enter date in dd/mm/yy format.

14/11/06

The difference between dates is 270

\*/

1. Write a program to accept date in dd-mm-yyyy format and print the same in words

Input : 31-03-2012

Output1 : Thirty one – Three – Two Thousand and Twelve

Output2 : 31st-March-2012

import java.io.\*;

public class DateToWords {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int dd, mm, yyyy;

String s;

DateToWords()

{

dd=mm=yyyy=0;

}

void Accept()throws IOException

{

System.out.println("Enter date in dd-mm-yyyy format.");

s=br.readLine();

dd=Integer.parseInt(s.substring(0,2));

mm=Integer.parseInt(s.substring(3,5));

yyyy=Integer.parseInt(s.substring(6));

}

void Display1()

{

String d1[]={"","One","Two","Three","Four","Five","Six","Seven","Eight","Nine"};

String d2[]={"Ten","Eleven","Twelve","Thirteen","Fourteen","Fifteen","Sixteen","Seventeen","Eighteen","Nineteen"};

String d3[]={"","","Twenty","Thirty","Forty","Fifty","Sixty","Seventy","Eighty","Ninety"};

String s="";

//Day

if(dd<=9)

s=s+d1[dd]+" - ";

else if(dd>9&&dd<=19)

s=s+d2[dd%10]+" - ";

else

s=s+d3[dd/10]+" "+d1[dd%10]+" - ";

//Month

if(mm<=9)

s=s+d1[mm]+" - ";

else

s=s+d2[mm%10]+" - ";

//Year

if(yyyy/100<=19)

{

s=s+d2[(yyyy/100)%10];

if(yyyy%100>=1&&yyyy%100<=9)

s=s+" Hundred and "+d1[yyyy%10];

else if(yyyy%100>9&&yyyy%100<=19)

s=s+" "+d2[yyyy%10];

else

s=s+" "+d3[(yyyy%100)/10];

}

else

{

s=s+d1[yyyy/1000]+" Thousand ";

if(yyyy%100>=1&&yyyy%100<=9)

s=s+" and "+d1[yyyy%10];

else if(yyyy%100>9&&yyyy%100<=19)

s=s+" and "+d2[yyyy%10];

else if(yyyy%100>19)

s=s+" and "+d3[(yyyy%100)/10];

}

System.out.println(s);

}

void Display2()

{

String s="";

s=s+dd;

//string for the end of day

String e[]={"","st","nd","rd","th","th","th","th","th","th"};

s=s+e[dd%10]+" - ";

String m[]={"","January","February","March","April","May","June","July","August","Semptember","October","November","December"};

s=s+m[mm]+" - "+yyyy;

System.out.println(s);

}

public static void main(String args[])throws IOException

{

DateToWords D=new DateToWords();

D.Accept();

D.Display1();

D.Display2();

}

}

/\*OUTPUT

Enter date in dd-mm-yyyy format.

31-03-2012

Thirty One - Three - Two Thousand and Twelve

31st - March - 2012

\*/

**Inheritance**

**Question 1(2012)**

A super class Detail has been defined to store the details of a customer. Define a sub class Bill to compute the monthly telephone charge of the customer as per the chart given below:

**Number of calls Rate**

* 1. Only rental charge
  2. 60 paise per call + rental charge
  3. 80 paise per call + rental charge

Above 300 1 rupee per call + rental charge

The details of both the classes are given below.

**Class name : Detail**

Data Members / Instance Variables

name : to store the name of the customer

address : to store the address of the customer

telno : to store the phone number of the customer

rent : to store the monthly rental charge

Member function

Detail(…) : parameterized constructor to assign

values to data members

void show() : to display the details of the customer

**Class name : Bill**

Data Members / Instance Variables:

n : to store the number of calls

amt : to store the amount to be paid by the customer

Member Functions:

Bill(…) : parameterized constructor to assign values

to data members of both classes and

initialize amt=0.0

void cal() : calculates the monthly telephone charge as

per the chart given above

void show() : displays the details of the customer and

amount to be paid

Specify the class Detail giving details of the constructor() and void show(). Using the concept of inheritance, specify the class Bill giving details of the constructor(), void cal, and void show.

THE MAIN() FUNCTION AND ALGORITHM NEED TO BE WRITTEN.

import java.io.\*;

public class Details{

String name, address;

long telno;

double rent;

Details(String n1, String a1, long t1, double r1)

{

name=n1;

address=a1;

telno=t1;

rent=r1;

}

void show()

{

System.out.println("Details of Customer");

System.out.println("Name: "+name);

System.out.println("Address: "+address);

System.out.println("Telephone Num: "+telno);

System.out.println("Rent: "+rent);

}

}

import java.io.\*;

public class Bill extends Details

{

int n;

double amt;

Bill(String n1, String a1, long t1, double r1, int n2, double amt2)

{

super(n1,a1,t1,r1);

n=n2;

amt=amt2;

}

void charge()

{

if(n>=1&&n<=100)

amt=rent;

else if(n>100&&n<=200)

amt=rent+0.60\*(n-100);

else if(n>200&&n<=300)

amt=rent+60+0.80\*(n-200);

else

amt=rent+60+80+(n-300);

}

void show()

{

super.show();

System.out.println("Number of Calls: "+n);

System.out.println("Amount: "+amt);

}

public static void main(String args[])

{

Bill B=new Bill("ABC","XYZ",98874,100,250,0.0);

B.charge();

B.show();

}

}

/\*OUTPUT

Details of Customer

Name: ABC

Address: XYZ

Telephone Num: 98874

Rent: 100.0

Number of Calls: 250

Amount: 200.0

\*/

**Recursion**

**Question 1**

A class HiFact has been defined to find the HCF of two numbers using recursive technique. This HCF is used to find the LCM of the two numbers. Some members of the class are given below:

Class Name: HiFact

Data Members: int a, b, hcf, lcm.

HiFact(): contructor to assign initial values

void getdata(): to input values of 'a' and 'b'

void change(): to swap a and b if a>b

int rechcf(int, int): to find hcf using recursive technique

int fn\_lcm(int, int, int,): to find lcm using 'a', 'b', and hcf

void results(): To invoke rechcf() and fn\_lcm() and to print lcm, hcf of the two numbers, 'a' and 'b'

import java.io.\*;

public class HiFact {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int a,b,hcf,lcm;

HiFact()

{

a=b=hcf=lcm=0;

}

void getData()throws IOException

{

System.out.println("Enter value for a");

a=Integer.parseInt(br.readLine());

System.out.println("Enter value for b");

b=Integer.parseInt(br.readLine());

}

void Change()

{

int t;

if(a>b)

{

t=a;

a=b;

b=t;

}

}

int rechcf(int x, int y)

{

if(y== 0)

return x;

else

return rechcf(y, x%y);

}

int fn\_lcm(int x, int y, int z)

{

return (x\*y)/z;

}

void results()

{

hcf=rechcf(a, b);

lcm=fn\_lcm(a, b, hcf);

System.out.println("The HCF of "+a+" and "+b+" = "+hcf);

System.out.println("The LCM of "+a+" and "+b+" = "+lcm);

}

public static void main(String args[])throws IOException

{

HiFact H=new HiFact();

H.getData();

H.Change();

H.results();

}

}

/\*OUTPUT

Enter value for a

20

Enter value for b

15

The HCF of 15 and 20 = 5

The LCM of 15 and 20 = 60

\*/

**Question 2**

Class name: Convert

Data Members: int n (integer whose digits are to be expressed in words)

Convert(): constructor to assign 0 to n

void inpnum(): to accept value of n

void extdigit(int): to extract digits of n using recursive technique

void num\_to\_words(int): to display digits of an integer in words

import java.io.\*;

public class Convert {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

int n;

Convert()

{

n=0;

}

void inpnum()throws IOException

{

System.out.println("Enter a number");

n=Integer.parseInt(br.readLine());

extdigit(n);

}

void extdigit(int x)

{

int r;

if(x==0)

return;

else

{

r=x%10;

extdigit(x/10);

num\_to\_words(r);

}

}

void num\_to\_words(int x)

{

String a[]={"zero","one","two","three","four","five","six","seven","eight","nine"};

String s="";

s=s+a[x]+" ";

System.out.print(s);

}

public static void main(String args[])throws IOException

{

Convert C=new Convert();

C.inpnum();

}

}

/\*OUTPUT

Enter a number

1357

one three five seven

\*/

**Linked Lists**

1. Generic list with insertBegin, insertEnd, insertMiddle, deletes, etc.
2. Count all the nodes in a list
3. Sum of all elements in a list
4. Search for a value in the list
5. Split a list into two
6. Merge two lists into one
7. Reverse a list
8. Sort a list
9. Compare two lists and check is they are equal
10. Implement stack using linked lists
11. Implement queue using linked lists

Merge as many of these functions as possible into one menu-driven program. Seperate programs for things like merging, splitting, stacks, queues, comparing can be used.

public class Node {

int data;

Node link;

Node()

{

link =null;

data=0;

}

Node(int d, Node l)

{

link =l;

data =d;

}

}

import java.io.\*;

public class Linked1 {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

Node start;

Linked1()

{

start=null;

}

void InsertB(int val)

{

Node nptr=new Node(val, null);

if(start==null)

start=nptr;

else

{

nptr.link=start;

start=nptr;

}

}

void InsertE(int val)

{

Node temp;

Node nptr=new Node(val, null);

if(start==null)

start=nptr;

else

{

temp=start;

while(temp.link!=null)

{

temp=temp.link;

}

temp.link=nptr;

}

}

void InsertM(int p, int val)

{

Node nptr=new Node(val, null);

Node temp=start;

int c=0;

while(temp.link!=null)

{

c=c+1;

if(c==p)

break;

temp=temp.link;

nptr.link=temp.link;

temp.link=nptr;

}

}

void Delete(int val)

{

Node temp =start;

int c=0;

if(start==null)

System.out.println("Not possible");

else

{

while(c<val-1)

{

temp=temp.link;

temp.link=temp.link.link;

c++;

}

}

}

int count(Node head)

{

if(head==null)

return 0;

else

return 1+count(head.link);

}

int sum(Node head)

{

if(head==null)

return 0;

else

return head.data+sum(head.link);

}

void Search(int val)

{

Node temp=start;

boolean f=false;

int c=0;

if(start==null)

f=false;

else

{

while(temp.link!=null)

{

if(temp.data==val)

{

f=true;

System.out.println("Found in node number "+c);

break;

}

else

{

temp=temp.link;

c=c+1;

}

}

if(f==false)

{

if(temp.data==val) //to check if it's in the last node

System.out.println("Found in node number "+(c+1));

else

System.out.println("No such number");

}

}

}

void Display()

{

Node temp;

temp=start;

while(temp.link!=null)

{

System.out.print(temp.data+" ");

temp=temp.link;

}

System.out.println(temp.data);

}

void Reverse()

{

Node nearNode, midNode, farNode;

nearNode=start;

midNode = nearNode.link;

farNode = midNode.link;

nearNode.link = null;

while(farNode!=null){

midNode.link = nearNode;

nearNode = midNode;

midNode = farNode;

farNode = farNode.link;

}

midNode.link = nearNode;

start = midNode;

}

void Sort()

{

int c=count(start);

Node temp;

int i,k;

int x;

for(i=0;i<=c-2;i++)

{

temp=start;

for(k=0;k<=c-2-i;k++)

{

while (temp.link != null) {

if (temp.data > temp.link.data) {

x = temp.data;

temp.data = temp.link.data;

temp.link.data = x;

}

temp = temp.link;

}

}

}

}

void Menu()throws IOException

{

int ch, num;

while(true)

{

System.out.println("1. Insert Begin");

System.out.println("2. Insert End");

System.out.println("3. Insert Middle");

System.out.println("4. Delete");

System.out.println("5. Count");

System.out.println("6. Sum");

System.out.println("7. Search");

System.out.println("8. Display");

System.out.println("9. Reverse");

System.out.println("10. Sort");

System.out.println("11. Exit");

System.out.println("Enter choice");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter number");

num=Integer.parseInt(br.readLine());

InsertB(num);

break;

case 2:

System.out.println("Enter number");

num=Integer.parseInt(br.readLine());

InsertE(num);

break;

case 3:

System.out.println("Enter number");

num=Integer.parseInt(br.readLine());

System.out.println("Enter place to insert");

int p=Integer.parseInt(br.readLine());

InsertM(p, num);

break;

case 4:

System.out.println("Enter the subscript whose data you want to delete");

num=Integer.parseInt(br.readLine());

Delete(num);

break;

case 5:

System.out.println("Number of nodes is "+count(start));

break;

case 6:

System.out.println("Sum is "+sum(start));

break;

case 7:

System.out.println("Enter number");

num=Integer.parseInt(br.readLine());

Search(num);

break;

case 8:

Display();

break;

case 9:

Reverse();

break;

case 10:

Sort();

break;

case 11:

System.exit(0);

break;

default:

System.out.println("Wrong Entry");

}

}

}

public static void main(String args[])throws IOException

{

Linked1 L=new Linked1();

L.Menu();

}

}

/\*OUTPUT

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

1

Enter number

10

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

2

Enter number

30

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

1

Enter number

40

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

2

Enter number

50

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

3

Enter number

44

Enter place to insert

2

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

4

Enter the subscript whose data you want to delete

2

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

5

Number of nodes is 4

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

6

Sum is 130

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

8

40 10 30 50

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

7

Enter number

30

Found in node number 2

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

9

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

8

50 30 10 40

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

10

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

8

10 30 40 50

1. Insert Begin

2. Insert End

3. Insert Middle

4. Delete

5. Count

6. Sum

7. Search

8. Display

9. Reverse

10. Sort

11. Exit

Enter choice

11

\*/

import java.io.\*;

public class Linked2 {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

Node start;

Linked2()

{

start=null;

}

void InsertB(int val)

{

Node nptr=new Node(val, null);

if(start==null)

start=nptr;

else

{

nptr.link=start;

start=nptr;

}

}

void Accept()throws IOException

{ int i,num;

for(i=0;i<3;i++)

{

System.out.println("Enter a number");

num=Integer.parseInt(br.readLine());

InsertB(num);

}

}

void Compare(Linked2 X)

{

boolean f=true;

Node temp1, temp2;

temp1=start;

temp2=X.start;

while(temp1.link!=null)

{

if(temp1.data!=temp2.data)

{

f=false;

break;

}

else

{

temp1=temp1.link;

temp2=temp2.link;

}

}

if(temp1.data!=temp2.data) //to compare last nodes

f=false;

if(f==false)

System.out.println("The lists are NOT equal.");

else

System.out.println("The lists are equal.");

}

void Merge(Linked2 X)

{

Node temp1,temp2;

temp1=start;

temp2=X.start;

while(temp1.link!=null)

{

temp1=temp1.link;

}

temp1.link=temp2;

}

int count(Node head)

{

if(head==null)

return 0;

else

return 1+count(head.link);

}

Linked2 Split()

{

Linked2 Z=new Linked2();

int i;

int c=count(start);

Node temp;

temp=start;

for(i=1;i<(c/2);i++)

{

temp=temp.link;

}

Z.start=temp.link;

temp.link=null;

return Z;

}

void Display()

{

Node temp;

temp=start;

while(temp.link!=null)

{

System.out.print(temp.data+" ");

temp=temp.link;

}

System.out.println(temp.data);

}

public static void main(String args[])throws IOException

{

Linked2 L1=new Linked2();

Linked2 L2=new Linked2();

Linked2 L3=new Linked2();

System.out.println("Enter the first three elements for List 1 and next three elements for List 2");

L1.Accept();

L2.Accept();

L1.Compare(L2);

L1.Merge(L2);

System.out.println("After lists are merged:");

L1.Display();

L3=L1.Split();

System.out.println("After list is split:");

L1.Display();

L3.Display();

}

}

/\*OUTPUT

Enter the first three elements for List 1 and next three elements for List 2

Enter a number

10

Enter a number

20

Enter a number

30

Enter a number

10

Enter a number

20

Enter a number

40

The lists are NOT equal.

After lists are merged:

30 20 10 40 20 10

After list is split:

30 20 10

40 20 10

\*/

import java.io.\*;

public class LinkedStack {

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

Node start;

LinkedStack()

{

start=null;

}

void Push(int val) //same as insertB

{

Node nptr=new Node(val, null);

if(start==null)

start=nptr;

else

{

nptr.link=start;

start=nptr;

}

}

void Pop()

{

if(start==null)

System.out.println("Stack Empty");

else

{

System.out.println("Popped value: "+start.data);

start=start.link;

}

}

void Display()

{

Node temp;

temp=start;

if(start==null)

System.out.println("Stack Empty");

else

{

while(temp.link!=null)

{

System.out.print(temp.data+" ");

temp=temp.link;

}

System.out.println(temp.data);

}

}

void Menu()throws IOException

{

int ch;

while(true)

{

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");

ch=Integer.parseInt(br.readLine());

switch(ch)

{

case 1:

System.out.println("Enter number");

int num=Integer.parseInt(br.readLine());

Push(num);

break;

case 2:

Pop();

break;

case 3:

Display();

break;

case 4:

System.exit(0);

break;

default:

System.out.println("Wrong entry");

}

}

}

public static void main(String[] args)throws IOException

{

LinkedStack L=new LinkedStack();

L.Menu();

}

}

/\*OUTPUT

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

10

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

20

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

30

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 30

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 20

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Popped value: 10

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

2

Stack Empty

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

50

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

1

Enter number

60

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

3

60 50

1. Push

2. Pop

3. Display

4. Exit

Enter your choice

4

\*/

import java.io.\*;

public class LinkedQ

{

BufferedReader br=new BufferedReader(new InputStreamReader (System.in));

Node start;

int r,f; //r=rear and f=front

final int max=5;

LinkedQ()

{

start=null;

f=0;r=-1;//r=-1

}

void Insert(int val) //same as InsertE

{

Node temp;

Node nptr=new Node(val, null);

if(start==null)

start=nptr;

else

{

temp=start;

while(temp.link!=null)

{

temp=temp.link;

}

temp.link=nptr;

}

}

void Delete()//same as Pop

{

if(start==null)

System.out.println("Queue Empty");

else

{

System.out.println("Deleted value: "+start.data);

start=start.link;

}

}

void Display()

{

Node temp;

temp=start;

if(start==null)

System.out.println("Queue Empty");

else

{

while(temp.link!=null)

{

System.out.print(temp.data+" ");

temp=temp.link;

}

System.out.println(temp.data);

}

}

void Menu()throws IOException

{

int n,num;

while(true)

{

System.out.println("1.Insert");

System.out.println("2.Delete");

System.out.println("3.Display");

System.out.println("4.Exit");

System.out.println("Enter your choice");

n=Integer.parseInt(br.readLine());

switch(n)

{

case 1:

System.out.println("Enter the number you wish to store on queue" );

num=Integer.parseInt(br.readLine());

Insert(num);

break;

case 2:

Delete();

break;

case 3:

Display();

break;

case 4:

System.exit(0);

default:

System.out.println("Wrong entry");

}

}

}

public static void main(String args[])throws IOException

{

LinkedQ Q=new LinkedQ();

Q.Menu();

}

}

/\*OUTPUT

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

67

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

78

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

1

Enter the number you wish to store on queue

89

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

3

67 78 89

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Deleted value: 67

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Deleted value: 78

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Deleted value: 89

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

2

Queue Empty

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice

4

\*/

**Assigment – Trees**

You will be required to create a menu-driven program for a binary tree which does the following:

1. Create and insert a node
2. Display inorder
3. Display preorder
4. Display postorder
5. Count the number of nodes and display
6. Search for a value in the tree

public class BinTreeNode {

BinTreeNode llink, rlink;

int data;

public BinTreeNode()

{

llink=rlink=null;

data=0;

}

public BinTreeNode(int n)

{

llink=rlink=null;

data=n;

}

}

import java.io.\*;

public class BinTree {

BinTreeNode root, nptr;

public BinTree()

{

root=null;

}

public void createNode(int val)

{

nptr=new BinTreeNode(val);

}

BinTreeNode InsertNode(BinTreeNode root, BinTreeNode NewNode)

{

if(root==null)

root=NewNode;

else if(NewNode.data<root.data)

root.llink=InsertNode(root.llink, NewNode);

else if(NewNode.data>root.data)

root.rlink=InsertNode(root.rlink, NewNode);

return root;

}

void inorder(BinTreeNode r)

{

if(r!=null)

{

inorder(r.llink);

System.out.println(r.data);

inorder(r.rlink);

}

}

void preorder(BinTreeNode r)

{

if(r!=null)

{

System.out.println(r.data);

inorder(r.llink);

inorder(r.rlink);

}

}

void postorder(BinTreeNode r)

{

if(r!=null)

{

inorder(r.llink);

inorder(r.rlink);

System.out.println(r.data);

}

}

int countNodes(BinTreeNode r)

{

if(r==null)

return 0;

else

{

int count=1;

count+=countNodes(r.llink);

count+=countNodes(r.rlink);

return count;

}

}

boolean search(BinTreeNode r, int val)

{

boolean f=false;

while((r!=null)&&!f)

{

int rval=r.data;

if(val<rval)

r=r.llink;

else if(val>rval)

r=r.rlink;

else

{

f=true;

break;

}

f=search(r,val);

}

return f;

}

void menu()throws IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int c,n;

while(true)

{

System.out.println("1. Insert Node");

System.out.println("2. Inorder");

System.out.println("3. Preorder");

System.out.println("4. Postorder");

System.out.println("5. Count Nodes");

System.out.println("6. Search for a node");

System.out.println("7. Exit");

System.out.println("Enter your choice.");

c=Integer.parseInt(br.readLine());

switch(c)

{

case 1:

System.out.println("Enter a number");

n=Integer.parseInt(br.readLine());

createNode(n);

root=InsertNode(root, nptr);

break;

case 2:

System.out.println("INORDER TREE TRAVERSAL");

inorder(root);

break;

case 3:

System.out.println("PREORDER TREE TRAVERSAL");

preorder(root);

break;

case 4:

System.out.println("POSTORDER TREE TRAVERSAL");

postorder(root);

break;

case 5:

System.out.println("Number of nodes is "+countNodes(root));

break;

case 6:

System.out.println("Enter a number");

n=Integer.parseInt(br.readLine());

if(search(root, n))

System.out.println("Found");

else

System.out.println("Not Found");

break;

case 7:

System.exit(0);

default:

System.out.println("Wrong Entry");

}

}

}

public static void main(String args[])throws IOException

{

BinTree B=new BinTree();

B.menu();

}

}

/\*OUTPUT

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

1

Enter a number

60

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

1

Enter a number

40

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

1

Enter a number

50

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

2

INORDER TREE TRAVERSAL

40

50

60

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

3

PREORDER TREE TRAVERSAL

60

40

50

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

4

POSTORDER TREE TRAVERSAL

40

50

60

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

5

Number of nodes is 3

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

6

Enter a number

50

Found

1. Insert Node

2. Inorder

3. Preorder

4. Postorder

5. Count Nodes

6. Search for a node

7. Exit

Enter your choice.

7

\*/