Good evening every one

welcome to durgasoft online training...

welcome to Data Structures and Algorithms with Java Batch:01

K Prakash Babu, 16 Years as Technical Trainer.

Data Structures ---->
Algorithms --------->
Language -----------> Java

http://www.durgasoft.com/DSAwithJAVA-Prakash-Online.asp

https://www.youtube.com/watch?v=1GjEBFdmpqA&list=PLd3UqWTnYXOnf4Vmn_mOlmzRzrTJo4ud2

Chapter: 01 Introduction to DSA

Chapter: 02 Problem Solving

Chapter: 03 Time and Space Complexity

Chapter: 04 Notations

Chapter: 05 Aspects of Algorithms

Chapter: 06 Mathematical Algorithms

Chapter: 07 Bitwise opertations

Chapter: 08 Recursion

Chapter: 09 Arrays

Chapter: 10 Matrix

Chapter: 11 Searching and Sorting Techniques

Chapter: 12 LinkedLists

Chapter: 13 Stacks (LIFO)

Chapter: 14 Queues

Chapter: 15 Binary Trees

Chapter: 16 Hashtables

Chapter: 17 Hashing

Chapter: 18 AVL Tree

Chapter: 19 Dictionaries or Maps

Chapter: 20 Graph

Chapter: 21 Greedy

Chapter: 22 Backtracking

Chapter: 23 Dynamic programming

duration ---> 2 to 3 months
timings ----> 7pm to 8pm [mon-fri]
fees -------> Rs. 999/-

7207212427
9246212143

core java batch is running at 7am morning ---> 6 class completed ---> Rs. 5000/-

Logic Based Programming -----> 8PM ---> 20/350 ---> C/Java/Python --> Rs. 2000/-

https://us02web.zoom.us/meeting/register/tZwkdeGtqz0iHNL6acB93gTQKRrg4IRpLNcs

https://www.youtube.com/watch?v=OG3gJxp6y_A&list=PLd3UqWTnYXOl1Qwbsqa3gDJieL3o9zmPP

https://www.youtube.com/watch?v=1GjEBFdmpqA&t=9s

Music Players ----> Jio Savan, Spotify, Prime Music etc
=============
To implement music player application, internally they are using data structures

data ---> .mp3 songs

searching, sorting, queue

Linked List ----> Music Players

Spam email detection ---> rxxxxxxxxx@gmail.com
====================

10 Unknown companies ---> 10 spam per day

throw 'xyz' ----> spam

will you just explain the data structure or do its coding also?

Yes, main target is T+C --> Coding with Java

String Data Structure ----> spam emails

abcdf@xyz.com ----> "xyz.com"

BookMyShow Application ---> movie ticket application

Array Data Structure ---> 2-D arrays ---> resevation of seats

sir actually we don't use the logic behind this DSA(we simply import) then why we have to learn.
(sir please don't mind hope this is a good question)

```
class MyLinkedList{
      ----------------
      ----------------
      ----------------
      ----------------
}

LinkedList LL = new LinkedList();
```

Data Structure:

Arranging data in a proper structure is called as Data Structures or Organizing Data is called as Data Structure.

The following are the various operations, that we can able to perform on data structures

1. inserting new data
2. removal of existing data
3. updating or replacement of data
4. searching (linar, binary)
5. sorting (15+ sorting)

etc

If we use above algorithms very effecively on data structures then so many factors will be improved

Student Data ----> read ---> ArrayList/Array
Student Data ----> insert -> LinkedList

Algorithm:

==> Step by step process to solve a problem is called as an algorithm.
==> Here we will concentrate on design part but not on implementation (Java).
==> no syntax, english instructions.

Problem Statement:

Implement a program to perform addition of two numbers.

Alg:

```
      1. read two numbers from the user.
      2. use arithmetic operators to calculate result/addition.
      3. c = a + b;
      4. print result to the console,screen,file or data base.
```

Implementation:

```
      int addition(int a,int b)
      {
            int c;
            c = a + b;
            return c;
      }
```

Ex:

```
class Demo
{
      static int addition(int a,int b)
      {
            int c;
            c = a + b;
            return c;
      }
}

class Test
{
      public static void main(String[] args)
      {
            System.out.println(Demo.addition(1,2));//3
            System.out.println(Demo.addition(2,3));//5
      }
}
```

```
C:\Users\redpr>cd\

C:\>cd prakashclasses

C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
3
5
```

Photos in Gallery
=================
==> Which DS followed in Gallery Application
==> Stack

Sending emails to the persons
=============================

==> 200 students
==> send a mail to all students
==> Data Structure: Queue

File Explorere | Directory Explorer | Folder Explorer

==> c:/prakashclasses/
==> Data Structure: Tree

GPS Navigation System

==> Travel from one location to another location
==> Data Structure: Graph

trending videos in youtube which data strucure is used sir?

Priority Queue

Q ---> V1(2M), V2(5M), V3(1M), V4(6M), V5(3M)

Sorting ---> V4, V2, V5, V1, V3

durgasoftonlinetraining@gmail.com
Bank name: ICICI
Account Holder's Name: DURGASOFT TRAINING SERVICES PVT LTD
Type : Current Account
Account No: 236305500267
IFSC Code: ICIC0002363
Hyderabad - 500038
Telangana

GOOGLE PAY/PHONE PAY NUMBER
9297972727

LBP vs DSAJ

LBP ---> Logical Thinking and Coding Skills
DSAJ --> Datastructures and Algorithms by using Java

PROBLEM STATEMENT:

Write a program to swap given two integer values.

Algorithm:

```
      ==> read two parameters from the user.
      ==> implement business logic related to swaping

            Logic1
            Logic2
            Logic3
            Logic4
            Logic5

      ==> print the data before swaping and after swaping
```

Note: In Java call by reference concept not threre.

Logic1:

```
       ==> declare one temp variable 'temp'
       ==> print a and b values
       ==>
            temp = a;
            a = b;
            b = temp;

       ==> print a and b value
```

Implementation:

```
import java.util.*;

class Demo
{
      static void swap(int a,int b)
      {
            System.out.println("before swaping a="+a+" and b="+b);
            int t;
            t = a;
            a = b;
            b = t;
            System.out.println("after swaping a="+a+" and b="+b);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            Demo.swap(a,b);
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value
10
Enter b value
12
```

```
before swaping a=10 and b=12
after swaping a=12 and b=10
```

Logic2:

```
       ==> print a and b values
       ==> by using addition and subtraction

            a = a + b;
            b = a - b;
            a = a - b;

       ==> print a and b value
```

Implementation:

```
import java.util.*;

class Demo
{
      static void swap(int a,int b)
      {
            System.out.println("before swaping a="+a+" and b="+b);
            a = a + b;
            b = a - b;
            a = a - b;
            System.out.println("after swaping a="+a+" and b="+b);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            Demo.swap(a,b);
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value
4
Enter b value
8
before swaping a=4 and b=8
after swaping a=8 and b=4
```

Logic3:

```
       ==> print a and b values
       ==> by using multiplication and division

            a = a * b;
            b = a / b;
            a = a / b;

       ==> print a and b value
```

Implementation:

```
import java.util.*;

class Demo
{
      static void swap(int a,int b)
      {
            System.out.println("before swaping a="+a+" and b="+b);
            a = a * b;
            b = a / b;
            a = a / b;
            System.out.println("after swaping a="+a+" and b="+b);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            Demo.swap(a,b);
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value
4
Enter b value
8
before swaping a=4 and b=8
after swaping a=8 and b=4
```

Algorithm:

==> Step by step process to solve a problem is called as an algorithm.
==> Here we will concentrate on design part but not on implementation (Java).
==> no syntax, english instructions.

Priori Analysis	Posteriori Analysis
1. Algorithm	1. Program
2. Indepedent of programming lang	2. Language dependent
3. H/W independent	3. H/W dependent
4. time and space complexities notations	4. time and space will be measured in bytes and sec

Logic1: using temp variable
Logic2: without temp variable using add and sub
Logic3: without temp variable using mul and div

Logic4: without using temp variable using bitwise operators

```
       ==> print a and b values
       ==> by using bitwise operators

            a = a ^ b;
            b = a ^ b;
            a = a ^ b;

       ==> print a and b value
```

Note: Dec into Bin and Bin into Dec

Implementation:

```
import java.util.*;

class Demo
{
      static void swap(int a,int b)
      {
            System.out.println("before swaping a="+a+" and b="+b);
            a = a ^ b;
            b = a ^ b;
            a = a ^ b;
            System.out.println("after swaping a="+a+" and b="+b);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            Demo.swap(a,b);
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value
11
Enter b value
22
before swaping a=11 and b=22
after swaping a=22 and b=11

C:\prakashclasses>java Test
Enter a value
-2
Enter b value
-3
before swaping a=-2 and b=-3
after swaping a=-3 and b=-2
```

Logic5: without using temp variable by using single line

```
       ==> print a and b values
       ==> by using single line

            a = (a+b) - (b=a);

       ==> print a and b value
```

Implementation:

```
import java.util.*;

class Demo
{
      static void swap(int a,int b)
      {
            System.out.println("before swaping a="+a+" and b="+b);
            a = a+b-(b=a);
            System.out.println("after swaping a="+a+" and b="+b);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            Demo.swap(a,b);
```

```
	}
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value
12
Enter b value
13
before swaping a=12 and b=13
after swaping a=13 and b=12
```

DESIGN A PROGRAM TO FIND MAX OF TWO NUMBERS:

~~~~~~

Algorithm:

```
	1. read two numbers from the user as a and b

	2. apply logic

	   version1:
		   conditional operator

		   (condition)?tb:fb

		   (a>b)?a:b
	   version2:
		   Math.max(a,b)

	3. print the result
```

can we check how the Math.max logic implemented internally --> abstraction

Implementation:

```
import java.util.*;

class Demo
{
	static int max_version1(int a,int b)
	{
		//manual code
		return (a>b)?a:b;
	}

	static int max_version2(int a,int b)
	{
		//predefined lib's
		return Math.max(a,b);
```
~~~~~~

```
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            System.out.println("max value from version1=
"+Demo.max_version1(a,b));
            System.out.println("max value from version2=
"+Demo.max_version2(a,b));
      }
}

DESIGN A PROGRAM TO FIND WHETHER THE GIVEN NUMBER IS EVEN OR ODD:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Algorithm:
----------

      1. read number 'n' value from user
      2. if n is divisible by 2 means 'even' else 'odd'

            if(n%2==0)
                  print even
            else
                  print odd
Implementation:
---------------
import java.util.*;

class Demo
{
      static String check_evenorodd(int n)
      {
            //manual code
            return (n%2==0)?"EVEN NUMBER":"ODD NUMBER";
      }

}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter n value");
            int n = obj.nextInt();
            System.out.println(Demo.check_evenorodd(n));
      }
```

```
}

C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value
34
EVEN NUMBER

C:\prakashclasses>java Test
Enter n value
55
ODD NUMBER
```

DESIGN A PROGRAM TO FIND SUM OF 'N' NATURAL NUMBERS:

Algorithm:

```
      1. read 'n' value from the user.
      2. apply the logic to find sum of n natural numbers

         Logic1:
               sum=0;
               for(i=1;i<=n;i++)
               {
                  sum=sum+i;
               }

         Logic2:
               int sum(int n)
               {
                  if(n==0)
                        return 1;
                  else
                        return n+sum(n-1);
               }
         Logic3:
                 math formula ---> n*(n+1)/2
      3. print result on the screen
```

Implementation:

```
import java.util.*;

class Demo
{
      static int sumofn_v1(int n)
      {
            //for loop
            int sum=0;
            for(int i=1;i<=n;i++)
            {
                  sum=sum+i;
            }
            return sum;
      }

      static int sumofn_v2(int n)
      {
            //recursion
            if(n==0)
                  return 0;
            else
                  return n+sumofn_v2(n-1);
      }

      static int sumofn_v3(int n)
      {
            //math formula
            return (n*(n+1))/2;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter n value");
            int n = obj.nextInt();
            System.out.println(Demo.sumofn_v1(n));
            System.out.println(Demo.sumofn_v2(n));
            System.out.println(Demo.sumofn_v3(n));
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value
5
15
15
15
```

IMPLEMENT A PROGRAM TO READ THREE NUMBERS FROM THE USER AND CAL MAX OF THREE NUMBERS

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~
```

Algorithm:

1. read a,b and c values from user.

2. apply the logic

   logic1: by using manual method

```
                 (a>b && a>c)?a:((b>c)?b:c)

          logic2:  by using max method in math

                 Math.max(Math.max(number1,number2),number3)

       3. print the result
```

```
Implementation:
---------------
```

```java
import java.util.*;

class Demo
{
      static int max1(int a,int b,int c)
      {
            return (a>b && a>c)?a:(b>c?b:c);
      }

      static int max2(int a,int b,int c)
      {
            return Math.max(Math.max(a,b),c);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value");
            int a = obj.nextInt();
            System.out.println("Enter b value");
            int b = obj.nextInt();
            System.out.println("Enter c value");
            int c = obj.nextInt();
            System.out.println(Demo.max1(a,b,c));
            System.out.println(Demo.max2(a,b,c));
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value
1
Enter b value
2
Enter c value
3
3
3
```

```
C:\prakashclasses>java Test
Enter a value
1
Enter b value
2
Enter c value
-3
2
2

C:\prakashclasses>java Test
Enter a value
1
Enter b value
-2
Enter c value
-3
1
1
```

IMPLEMENT A PROGRAM TO FIND FACTORIAL OF THE GIVEN NUMBER
~~~~~~

Algorithm:

```
        1. read n value from the user.

        2. apply any one of the following logic

        logic1:
              by using looping

              f=1;
              for(i=1;i<=n;i++)
              {
                  f=f*i;
              }
              print f

        logic2:

              by using recursion

              int fact(int n)
              {
                  if(n==1)
                        return 1;
                  else
                        return n*fact(n-1);
              }

        3. print the result
```

IMPLEMENTATION:

~~~~~~

```
class Demo
{
      static int fact1(int n)
      {
            //loop
            int i,f=1;
            for(i=1;i<=n;i++)
            {
                  f=f*i;
            }
            return f;
      }

      static int fact2(int n)
      {
            //recursion
            if(n==1 || n==0)
                  return 1;
            else
                  return n*fact2(n-1);
      }
}

class Test
{
      public static void main(String[] args)
      {
            java.util.Scanner obj = new java.util.Scanner(System.in);
            System.out.println("Enter n value");
            int n = obj.nextInt();
            if(n>=0)
            {
            System.out.println("factorial by using loop = 
"+Demo.fact1(n));
            System.out.println("factorial by using recursion = 
"+Demo.fact2(n));
            }
            else
                  System.out.println("Arey what happend to you factorial 
for -ve num not existed");
      }
}

C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value
0
factorial by using loop = 1
factorial by using recursion = 1

C:\prakashclasses>java Test
Enter n value
-9
```

Arey what happend to you factorial for -ve num not existed

```
C:\prakashclasses>
```

IMPLEMENT A PROGRAM TO CHECK WHETHER THE GIVEN NUMBER IS PRIME OR NOT

~~~~~~

Algorithm:

```
        1. read a number 'n' from the user.

        2. apply logic

        logic1:
              by using loop

              factors = 0
              for(i=1;i<=n;i++)
              {
                  if(n%i==0)
                     factors++;
              }
              if factors==2 then print "Yes" else "No"

        logic2:
                by using recursion

              boolean isPrime(int n,int i) //i=n/2
              {
                  if(i==1)
                     return true;
                  else if(n%i==0)
                     return false;
                  else
                     return isPrime(n,--i);
              }

        3. print the status "Yes" or "No"
```

implementation:

```
class Demo
{
      static boolean isPrime1(int n)
      {
            //loop
            int i,f=0;
            for(i=1;i<=n;i++)
            {
                  if(n%i==0)
                        f++;
            }
            return f==2;
      }
```
~~~~~~

```
      static boolean isPrime2(int n,int i)
      {
            //recursion
            if(i==1)
                  return true;
            else if(n%i==0)
                  return false;
            else
                  return isPrime2(n,--i);
      }
}

class Test
{
      public static void main(String[] args)
      {
            java.util.Scanner obj = new java.util.Scanner(System.in);
            System.out.println("Enter n value");
            int n = obj.nextInt();
            for(int i=2;i<=n;i++)
            {
System.out.println(i+"\t"+(Demo.isPrime1(i)?"Yes":"No")+"\t"+(Demo.isPrime2(i,i/2)?"Yes":"No"));
            }
      }
}
```

IMPLEMENT A PROGRAM/ALG TO GENERATE FIBNOCCI NUMBERS

sum of previous two numbers, where the series starts from 0,1

```
      0, 1  ----> 0, 1, 1, 2, 3, 5, 8, 13, .....
```

algorithm:

1. read n value from the user.
2. Create Array List object
3. push all the calcualte fib seq, into array list

   logic:

```
         a = 0;
         b = 1;
         obj.add(a);
         obj.add(b);
         for(i=1;i<=n-2;i++)
         {
            c=a+b;
            obj.add(c);
            a=b;
            b=c;
         }
```

```
      4. print array list
```

```
implementation:
---------------
import java.util.*;

class Demo
{
      static ArrayList<Integer> getFibonacciNums(int n)
      {
            int a,b,c,i;
            ArrayList<Integer> al = new ArrayList<Integer>();
            a = 0;
            b = 1;
            al.add(a);
            al.add(b);
            for(i=1;i<=n-2;i++)
            {
                  c=a+b;
                  al.add(c);
                  a=b;
                  b=c;
            }
            return  al;
      }
}

class Test
{
      public static void main(String[] args)
      {
            java.util.Scanner obj = new java.util.Scanner(System.in);
            System.out.println("Enter n value");
            int n = obj.nextInt();
            System.out.println(Demo.getFibonacciNums(n));
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value
5
[0, 1, 1, 2, 3]

C:\prakashclasses>java Test
Enter n value
10
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

IMPLEMENT A PROGRAM/ALG TO GENERATE TRIBONACCI NUMBERS

----
sum of previous three numbers, where the series starts from 0,1,2

```
      0, 1, 2  ----> 0, 1, 2, 3, 6, 11, 20, ....
```

algorithm:

```
      1. read n value from the user.
      2. Create Array List object
      3. push all the calcualte trib seq, into array list

         logic:

         a = 0;
         b = 1;
         c = 2;
         obj.add(a);
         obj.add(b);
         obj.add(c);
         for(i=1;i<=n-3;i++)
         {
            d=a+b+c;
            obj.add(d);
            a=b;
            b=c;
            c=d;
         }

      4. print array list
```

implementation:

```
import java.util.*;

class Demo
{
      static ArrayList<Integer> getTribonacciNums(int n)
      {
            int a,b,c,d,i;
            ArrayList<Integer> al = new ArrayList<Integer>();
            a = 0;
            b = 1;
            c = 2;
            al.add(a);
            al.add(b);
            al.add(c);
            for(i=1;i<=n-3;i++)
            {
                  d=a+b+c;
                  al.add(d);
                  a=b;
                  b=c;
                  c=d;
            }
            return  al;
      }
}

class Test
{
      public static void main(String[] args)
      {
            java.util.Scanner obj = new java.util.Scanner(System.in);
            System.out.println("Enter n value");
            int n = obj.nextInt();
            System.out.println(Demo.getTribonacciNums(n));
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value
10
[0, 1, 2, 3, 6, 11, 20, 37, 68, 125]
```

performance of an algorithm

we can measure performance of an algorithm by using the following two components.

1. space complexity
2. time complexity

space complexity:

=> the space complexity of an algorithm is the amount of memory, it needs to run to complete task.

=> space complexity of any algorithm is calculate as,

```
      s(p) = fixed_part + variable_part
```

```
=> fixed part --> independent of instance characterstics
            --> space for variables, space for constants etc

=> variable_part --> dependent of instance characterstics
                 --> looping variables, arrays etc
```

Ex: addition of three numbers.

algorithm addition(a,b,c)

```
{
      return a+b+c;
}
```

```
space complexity ----->
                 a ---> 1 unit
                 b ---> 1 unit
                 c ---> 1 unit
                 -------------
                 total: 3 units
                 --------------
```

sp(addition) = 3 units

```
Ex: area of circle
------------------
algorithm areaofcircle(raidus)
{
      result = 3.147*radius*radius;
      return result;
}
```

```
space complexity ----->
                 radius ---> 1 unit
                 3.147  ---> 1 unit
                 result ---> 1 unit
                 ------------------
                 total: 3 units
                 ------------------
```

sp(addition) = 3 units

```
Ex: area of circle
------------------
algorithm areaofcircle(raidus)
{
      return 3.147*radius*radius;
}
```

```
space complexity ----->
                 radius ---> 1 unit
                 3.147  ---> 1 unit
                 ------------------
                 total: 2 units
                 ------------------
```

sp(addition) = 2 units

bits/bytes/kb/mb/gb/tb/pb etc

2 x 4 bytes = 8 bytes

```
int a = 5;

sp(alg) = 1 unit


Ex:

String s = "java";

sp(s) = 1 unit

c---> 1
py -> 1
java-> 1 unit

Ex:

int i = null;

1 unit

Ex: sum of 'n' natural numbers
------------------------------

algorithm sum_of_n(n)
{
      s=0;
      for(i=0;i<=n;i++)
      {
            s=s+i;
      }
      return s;
}

space complexity ------>

                  n ----> 1 unit
                  s ----> 1 unit
                  i ----> 1 unit
                  --------------
                  total-> 3 units
                  ---------------

Ex: sum of elements in an array 'a'
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
algorithm sum_array(a,n)
{
      s=0;
      for(i=0;i<n;i++)
      {
            s=s+a[i];
      }
      return s;
```

```
}


space complexity ------>
                 a ----> n units
                 n ----> 1 unit
                 s ----> 1 unit
                 i ----> 1 unit
                 -----------------
                 sp ---> n+3 units
                 ------------------
```

```
Ex: sum of elements present in an array by using recursion
----------------------------------------------------------
algorithm sum_of_elements_recursion(a,n)
{
      if(n<0)
            return 0;
      else
            return sum_of_elements_recursion(a,n-2)+a[n-1];
}

space complexity ----->
                 Rsum(a,n) -----> 1(a[n-1]) + 1(n) + 1(return) ---> 3
units
                 Rsum(a,n-1) ---> 1(a[n-2]) + 1(n) + 1(return) ---> 3
units
                 .
                 .
                 .
                 .
                 Rsum(a,n-n) --> 1(a[n-n]) + 1(n) + 1(return) ----> 3
units
                 ---------------------------------------------------------
---
                 total space complexity ----> 3(n+1) ===> 3n+3
                 ---------------------------------------------------------
---
```

```
Ex: Factorial of the given number using recursion
-------------------------------------------------

algorithm fact(n)
{
      if(n==0)
            return 1;
      else
            return n*fact(n-1);
}
```

```
space complexicity ------>
                       f(n) = 1 + 1
                       f(n-1) = 1 + 1
                       f(n-2) = 1 + 1
                       .
                       .
                       f(n-(n-1)) = 1 + 1
                       f(n-n) = 1
                       --------------------
                       sp(fact) = 2n+1 units
                       ---------------------
```

Ex: space complexity for prime number or not application using recursion

```
algorithm isprime(n,i)
{
      if(i==1)
            return true;
      else if(n%i==0)
            return false;
      else
            return isprime(n,--i);
}
```

```
sp(isprime) ----->
               isprime(n) -----> 2
               isprime(n-1) -----> 2
               isprime(n-2) -----> 2
               isprime(n-3) -----> 2
               isprime(n-n) -----> 1 or 1
               ---------------------------
               sp(isprime) = 2n+1
               ---------------------------
```

time complexity:

the time complexity of an algorithm is the amount of computer time it needs to complete the task.

```
tc(p) = compile time + execution time
      = execution time (ignore compile time, compilation will be done
only one time)
```

step count method to calculate time complexity

1) for algorithm heading ------> 0
2) for braces -----------------> 0
3) for expressions ------------> 1
4) for if conditions ----------> 1
5) for loops ------------------> based on number of iterations 'n'

java code for sorting ----> 10 lines

```
py code ------------------> 1 line L.sort()

Ex: addition of three numbers
------------------------------
1: algorithm addition(a,b,c)
2: {
3:    return a+b+c;
4: }

1 -----> 0
2 -----> 0
3 -----> 1 unit
4 -----> 0

tc(addition) = 1 unit

case1:

algorithm addition(a,b,c)
{
      int d = a+b+c;
      return d;
}

case2:

algorithm addition(a,b,c)
{
      return a+b+c;
}

Ex: find max of two numbers
---------------------------
1: algorithm max(a,b)
2: {
3:    return (a>b)?a:b;
4: }

1 -----> 0
2 -----> 0
3 -----> 1 unit
4 -----> 0

tc(addition) = 1 unit

Ex: find max of two numbers
---------------------------
1: algorithm max(a,b)
2: {
3:    if(a>b)
4:       return a;
5:    else
6:       return b;
7: }
```

```
1 -----> 0
2 -----> 0
3 -----> 1 unit
4 -----> 0
5 -----> 0
6 -----> 0
7 -----> 0
```

tc(addition) = 1 unit

```
Ex: find max of three numbers
---------------------------
1: algorithm max(a,b,c)
2: {
3:    if(a>b && a>c)
4:       return a;
5:    else if(b>a && b>c)
6:       return b;
7:    else
8:       return c;
9: }
```

```
1 -----> 0
2 -----> 0
3 -----> 1 unit
4 -----> 0
5 -----> 1 unit
6 -----> 0
7 -----> 0
8 -----> 0
9 -----> 0
```

tc(addition) = 2 unit

Note: &&, || operators are also called shortcircuit operators

cond1 && cond2 && cond3 && cond4 && cond5 ---> if first cond is false, then stop exe
cond1 || cond2 || cond3 || cond4 || cond5 ---> if first cond is true, continue

SUCH BEUTIFUL NATURE THEY HAVE DESIGNED AT LANGUAGE LEVEL

```
Ex: sum of 'n' natural numbers
------------------------------
1: algorithm sum(n)
2: {
3:    sum=0;
4:    for(i=1;i<=n;i++)
5:    {
6:        sum=sum+i;
7:    }
8:    return sum;
9: }
```

```
1 ------> 0
2 ------> 0
3 ------> 0
4 ------> n+1 ['n' times true '1' time false]
5 ------> 0
6 ------> n
7 ------> 0
8 ------> 0
9 ------> 0
```

tc(sum) = n+1+n = 2n+1

Ex: sum of 'n' even numbers

```
1: algorithm sum(n)
2: {
3:    sum=0;
4:    for(i=1;i<=n;i++)
5:    {
6:        if(i%2==0)
7:          sum=sum+i;
8:    }
9:    return sum;
10: }
```

```
1------> 0
2------> 0
3------> 0
4 -----> n+1
5 -----> 0
6 -----> n
7 -----> n/2
8 -----> 0
9 -----> 0
10 ----> 0
```

tc(sum) = n/2+2n+1

Ex: find max element present in an array

```
1.    algorithm maxElement(a,n)
2.  {
3.     max = a[0];
4.     for(i=1;i<n;i++)
5.       {
6.         if(max<a[i])
7.         {
8.                  max = a[i];
9.         }
10.    }
11.    return max;
12. }
```

```
1-----> 0
2 ----> 0
3 ----> 0
4 ----> n+1-1 = n  [1 comp we are not considering first element]
5 ----> 0
6 ----> n-1
7 ----> 0
8 ----> n-1
9 ----> 0
10 ---> 0
11 ---> 0
12 ---> 0
```

tc(maxElement) = n+n-1+n-1=3n-2

O(n)

Ex1: sorting of an array

```
algorithm sort(a,n)
{
      for(i=0;i<n;i++) ---------> n+1
      {
            for(j=i+1;j<n;j++) ---> n*(n) ---> n^2
            {
                  if(a[i]>a[j]) ----> n^2 - 1
                  {
                        t=a[i];
                        a[i]=a[j];
                        a[j]=t;
                  }
            }
      }
}
```

ts(sort) -----> n+1+n2+n2-1 ----> 2n^2+n

```
if data is already in sorting ASC ----> x
if data is already in sorting DESC ---> x
if data is not in sorting     --------> x
```

Ex2: sum of two matrices

```
algorithm(a,b,n,n)
{
      for(i=0;i<n;i++) --------> n+1
      {
            for(j=0;j<n;j++) ----> n*(n+1) ---> n^2 + n
            {
                  c[i][j] = a[i][j] + b[i][j]; ---> n^2
            }
      }
}
```

tc(sum of two mat) = n+1+n2+n+n2 ---> 2n^2 + 2n + 1 ---> O(n2)

Sir please explain time complexity of following cases

```
for(i=1;i<=n;i++) ----> n+1

Case 1
for(int i=0; i<=n; i++) -----> n+1+1 ---> n+2
Case 2
for(int i=0; i<n; i++) ------> n+1-1+1 -> n+1
Case 3
for(int i=1; i<=n; i++) ------> n+1
Case 4
for(int i=1; i<n; i++) -------> n+1-1 ---> n
```

Ex: Matrix Multiplication

```
for(i=0;i<n;i++) -------------------------------------> n+1+1-1===> n+1
{
      for(j=0;j<n;j++) ---------------------------------> n*(n+1+1-1) ==> n*(n+1) = n^2+n
      {
            c[i][j] = 0; ---------------------------------> n*(n) = n^2
            for(k=0;k<n;k++) -----------------------------> n^2 * (n+1-1+1) ==> n^2*(n+1)
            {
                  c[i][j] = c[i][j] + (a[i][k] * b[k][j]); --> n^2 * (n) ===> n^2*n=n^3
            }
      }
}
```

```
tc(mm) = n+1 + n^2+n + n^2 + n^3 + n^2 + n^3
       = 2n3 + 3n2 + 2n + 1
```

O(n3)

space and time complexity of any algorithm

algorithm cases   ----> best case, avg case, worst case
asymptotic notations --> 5 notations

introduction to algorithms
introduction to data structures
applications of data structures

steps to prepare algorithm
steps to prepare flowchart
steps to implement a program in java
sample programs (10 programs)
analysis of algorithms
space complexity calculation
time complexity calculation

best case, worst case and average case analysis

best case:

if we are looking for a sol, which is avaialble at very first location, then such type of case is called as best case.

Ex:

11, 12, 13, 14, 15, 16

key = 11

comp ---> 1st comp ---> best case

worst case:

if we are looking for a data, which is available at last position or may not be available, then such type of cases are called as worst case.

Ex:

11, 12, 13, 14, 15, 16

key = 16

comp ---> 6, success

key = 17

comp ---> 6, failure

average case:

if we are looking for multiple data's, the time/space taken for that alg is calcualted based on sum of the possible case.

Ex:

11, 12, 13, 14, 15, 16

key=11 ----> 1
key=12 ----> 2
key=13 ----> 3
key=14 ----> 4
key=15 ----> 5
key=16 ----> 6

key=17 ----> 6

avg case ===> total comp/num.of cases

Note: we can ignore this case.

Real time example:

i have given Rs. 10,000/- to my friend....

1st of everymonth -----> 50,000/-
15th of everymonth ----> 5000/-
30th of everymonth ----> 1000/-

1) 1st nov 2022 ----> Happy ----> Best case
2) 14th nov 2022 ---> Good -----> Average case
3) 16th nov 2022 ---> Good -----> Average case
4) end of the month-> Good -----> Worst case

Note: we can ignore this average case.

best case ------> 1 unit (constant value) ----> O(1)

average case ---> n units ------n-2--------------> O(n)

worst case ------> n units -----n--------------> O(n)

Ex: find max element present in an array

```
  algorithm maxElement(a,n)
{
   max = a[0];
   for(i=1;i<n;i++)
     {
        if(max<a[i])
        {
                max = a[i];
        }
   }
   return max;
}
```

11, 10, 9, 8, 7, 6 ----> max: 11

Hence, we will calcualte time and space complexity based on only WORST CASE COMP.

O(----)

O(1), O(n), O(n2), O(n3), O(logn), O(nlogn) etc

Asymptotic Notations:

it is used to measure/represent time and space complexity of any algorithm.

1. Big-Oh ----> O
2. Omega ----> W
3. Theta -----> theta
4. Little oh -> o
5. Little omega --> w little omega

Big "Oh" (O):-

a function f(n) is said to be in O(g(n)) denoted by f(n)=O(g(n)) is bounded above by some constant multiple of g(n) for all n, i.e. there exist positive constant 'c' and non-negative integer 'n0' such that f(n)<=c*g(n) for every n>=n0.

diagram

Ex:

f(n) = 2n+2
g(n) = n^2

where n>=3

Omega Notation (W):-

a function f(n) is said to be in W(g(n)) denoted by f(n)=W(g(n)) is bounded below by some constant multiple of g(n) for all n, i.e. there exist positive constant 'c' and non-negative integer 'n0' such that f(n)>=c*g(n) for every n>=n0.

diagram

Ex:

f(n) = 2n^2 + 3
g(n) = 7n

where n0>=3

Theta notation (0):-

a function f(n) is said to be in 0(g(n)) denoted by f(n)=0(g(n)) is bounded with above and below by some constant multiples of g(n) for all n, i.e. there exist positive constant 'c1' and 'c2' and non-negative integer 'n0' such that c1*g(n)<=f(n)<=c2*g(n) for every n>=n0.

digram

Ex:

f(n) = 4n+1
g(n) = n , c1=4 and c2 = 5

where n>=1

algorithm-sample algorithms
flow chart
implementing problems in java

analysis of algorithms
space complexity
time complexity
performance measurements (notations)

5pm ---> 7pm
----
algorithm-sample algorithms
flow chart
implementing problems in java

pending
----
analysis of algorithms
space complexity
time complexity
performance measurements (notations)

Recursion:
----
01) Introduction to functions
02) Why Recursion?
03) Recursion
04) Base condition
05) Finate and Infinate Recursion
06) Mathematical Interpretation of Recursion
07) Properties of Recursion
08) Advantages & disadvantages of recursion
09) Difference between iteration and recursion
10) Implement a program to print natural numbers from 1 to n
11) Implement a program to calculate sum of 'n' natural numbers
12) Implement a program to calculate a^b (a to the power b)
13) Implement a program to find factorial of the given number?
14) Implement a program to calculate product of two integer values (a*b)
15) Implement a program to check whether the given number is prime number or not?
16) Implement a program to find sum of digits present in the given number?
17) Implement a program to calcualte reverse of the given number?

18) Implement a program to count number of digits present in the given number?
19) Implement a program to convert decimal number into binary?
20) Implement a program to find nth fib number
21) Implement a program to find LCM of two numbers?
22) Implement a program to find HCF/GCD of the given two numbers
23) Implement a program to find reverse of the given string using recursion?
24) Implement a program to remove the given character from a string?
25) Implement a program to return Str, where all the adjacent chars are sep by a "*".
26) Implement a program to return new string where identical adjcent chars are sep by *
27) Implement a program to return true if a string nesting of zero or more pairs of ()
28) Implement a program to count number of times, the give char occurred.
29) IMP to replace the given old character with new character in the original string?
30) IMP to count the number of times given string appeared in the original string?
31) IMP to replace the given string with new string?
32) Towers of Hanoi

Recursion:-

==> function: set of instructions or sequence of operations under a common name or block

Ex:

```
	fun()
	{
		--------------
		--------------
		--------------
	}
```

we can call this fun any number of times based on our requirement.

advantage: ----> code reusability

Ex:

```
	int add(int a,int b){
		return a+b;
	}
```

Ex:

```
	boolean insertRecordInToMysqlDatabase(String name, int htno, double percentage){
			-----------------------
			-----------------------
			-----------------------
			-----------------------
	}
```

```
System.out.println(insertRecordInToMysqlDatabase("AAA",111,67.89));
System.out.println(insertRecordInToMysqlDatabase("BBB",222,77.89));
System.out.println(insertRecordInToMysqlDatabase("CCC",333,87.89));
System.out.println(insertRecordInToMysqlDatabase("DDD",444,97.89));
```

4+4+4+4=16
4+4=8

C/C++/Python ======> functions
Java/Python =======> methods

Ex:

```
      class Demo
      {
            boolean insertRecordInToMysqlDatabase(String name, int htno,
double percentage){
                  -----------------------
                  -----------------------
                  -----------------------
                  -----------------------
            }
      }

Demo obj = new Demo();
System.out.println(obj.insertRecordInToMysqlDatabase("AAA",111,67.89));
System.out.println(obj.insertRecordInToMysqlDatabase("BBB",222,77.89));
System.out.println(obj.insertRecordInToMysqlDatabase("CCC",333,87.89));
System.out.println(obj.insertRecordInToMysqlDatabase("DDD",444,97.89));
```

Q) can we declare a function within another function or not?

Yes, we can define, but only few programming languages are supporting this.

C -----> Yes
C++ ---> Yes
Java --> No

we can use a function within another function, we can't declare a definition of a function within another function/method.

Ex:

```
            void method1(){
                  ------------------
                  ------------------
                  void method2(){
                       -------------------
                       -------------------
                  }
                  -------------------
                  -------------------
            }
```

Ex:

```
          void method1(){
                  ------------------
                  ------------------
                  method2();
                  -------------------
                  -------------------
            }
```

Ex:

```
            void method1(){
                  ------------------
                  ------------------
                  Math.max(10,20);
                  -------------------
                  -------------------
            }
```

Recursion is a process of calling a method/function by itself, in this process the method which is invoked is called as 'Recursive Method'.

this recursion is divided into two ways based on method calls...

1) infinate recursion
2) finate recursion

infinate recursion:
----

the method which called by itself, infinate times. we will get Error message 'Stack Over Flow' error we will get.

Ex:

```
import java.util.*;
class Demo{
      void m(){
            System.out.println("Good Evening");
            m();
      }
}
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            Demo d = new Demo();
            d.m();
      }
}
```

output:

```
Good Evening
Good Evening
Good Evening
Good Evening
.
.
.
Exception in thread "main" java.lang.
```

finate recursion:-
----

a method which is called by itself, and terminates at finate number of steps is called as finate recursion.

we can make this finate recursion based on 'BASE CONDITION'.

base condition:

It is a special, we have to create inside recursive calls so that our recursion should terminate at a finate steps.

```
import java.util.*;
class Demo{
      static int c;
      void m(){
            if(c>10)
                  return;
            else
            {
                  System.out.println("Good Evening, c="+c);
                  c++;
                  m();
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            Demo d = new Demo();
            d.m();
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Good Evening, c=0
Good Evening, c=1
Good Evening, c=2
Good Evening, c=3
Good Evening, c=4
Good Evening, c=5
Good Evening, c=6
Good Evening, c=7
Good Evening, c=8
Good Evening, c=9
Good Evening, c=10
```

Why we need Recursion:

some tims, if we got a problem, we need to divide that big problems, into small small units, and find solutions for these sub-problems, which inturn creates solution for that bigger problem. This is the senario major applications are using. Ex: Recusrion, DAC...

Mathematical Interpretation of Recursion:

Ex: Sum of 'n' natural numbers

```
In Math:
           f(n) = 1 + 2 + 3 + 4 + 5 + ...... + n

In Recursion:

           f(n) = 1         , n=1
           f(n) = n + f(n-1), n>1

f(5) = n + f(n-1) = 5 + f(4)
f(4) = n + f(n-1) = 4 + f(3)
f(3) = n + f(n-1) = 3 + f(2)
f(2) = n + f(n-1) = 2 + f(1)
f(1) = 1

stack ===> 5, 4, 3, 2 , 1

1+2=3+3=6+4=10+5=15
```

Properties of Recursion:

1) same operations with multiple inputs.
2) we will divide the entire problem into small problems.
3) base condition is very very important in recursion, else it leads to infinate exe.

advantages of recursion:

1) recursive algorithms are easier to write.
2) easy to solve natural big problems, Ex: Towers of Hanoi problem
3) reduce unnecessary function calls.
4) reduce length of the code.
5) very useful while solving data structure related problems.

6) we can evaulate some expressions, infix, prefix and postfix etc

disadvanatges of recursion:

1) recursion uses extra stack space.
2) redundent computations
3) tracing will be difficult
4) slower in execution
5) runs out of memory (StackOverFlow Error)

sir what i remember is .....
java has garbage collectors which clears memory after execution.
......
so why we need to worry about space complexity at all

difference between recursion and iteration?

Recursion:

1) terminates when base condition is true.
2) functions concept.
3) extra space is required.
4) smaller code.

Iteration:

1) terminates when condition is false.
2) looping statement concepts.
3) extra space is not required.
4) bigger code.

01. Implement a program to print natural numbers from 1 to n

```
import java.util.*;
class Demo
{
      static void print(int n){
            if(n>=1)
            {
                  //System.out.print(n+" "); ===> n, n-1, n-2, ... 1
                  print(n-1);
                  System.out.print(n+" "); // ==> 1, 2, 3, 4, .... n
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int n = obj.nextInt();
            Demo.print(n);
      }
}


output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
10
1 2 3 4 5 6 7 8 9 10

02. Implement a program to calculate sum of 'n' natural numbers
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Demo
{
      static int sum(int n){
            if(n==1)
                  return 1;
            else
                  return n+sum(n-1);
      }
}
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int n = obj.nextInt();
            System.out.println(Demo.sum(n));
      }
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
5
15

C:\prakashclasses>java Test
4
10

03. Implement a program to calculate a^b (a to the power b)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Demo
{
      static int power(int a,int b){
```

```
            if(b>=1){
                  return a*power(a,b-1);
            }
            else
                  return 1;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int a = obj.nextInt();
            int b = obj.nextInt();
            System.out.println(Demo.power(a,b));
      }
}


output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
2
3
8

C:\prakashclasses>java Test
3
3
27

C:\prakashclasses>java Test
5
3
125

C:\prakashclasses>java Test
5
10
9765625
```

4. Implement a program to find factorial of the given number?

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Demo
{
      static int fact(int n){
            if(n==0)
                  return 1;
            else
                  return n*fact(n-1);
```

```
      }
}
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int n = obj.nextInt();
            System.out.println(Demo.fact(n));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
5
120

C:\prakashclasses>java Test
4
24
```

```
Ex:
      for(i=0;i<=10;i++){} ----> finate ===> 11 times
      for(i=0;i>=0;i++){} -----> infinate
```

05. Implement a program to calculate product of two integer values (a*b)

```
Ex:
import java.util.*;

class Demo
{
      static int product(int a,int b){
            if(a<b)
                  return product(b,a);
            else if(b!=0)
                  return a+product(a,b-1);
            else
                  return 0;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value:");
            int a = obj.nextInt();
            System.out.println("Enter b value:");
            int b = obj.nextInt();
            System.out.println(Demo.product(a,b));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter a value:
2
Enter b value:
3
6

C:\prakashclasses>java Test
Enter a value:
8
Enter b value:
8
64
```

06. Implement a program to check whether the given number is prime number or not?

~~~~~~

```java
import java.util.*;

class Demo
{
      static boolean isprime(int n,int i)
      {
            if(i==1)
                  return true;
            else if(n%i==0)
                  return false;
            else
                  return isprime(n,--i);

      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter n value:");
            int n = obj.nextInt();
            System.out.println(Demo.isprime(n,n/2));//true or false
      }
}
```
~~~~~~

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value:
2
true

C:\prakashclasses>java Test
Enter n value:
3
true

C:\prakashclasses>java Test
Enter n value:
4
false

C:\prakashclasses>java Test
Enter n value:
5
true

C:\prakashclasses>java Test
Enter n value:
6
false
```

07. Implement a program to find sum of digits present in the given number?

```
------------------------------------------------------------------------
-
import java.util.*;

class Demo
{
      static int sumofdigits(int n)
      {
            if(n==0)
                  return 0;
            else
                  return (n%10)+sumofdigits(n/10);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter n value:");
            int n = obj.nextInt();
            System.out.println(Demo.sumofdigits(n));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n value:
5
5

C:\prakashclasses>java Test
Enter n value:
123
6

C:\prakashclasses>java Test
Enter n value:
1234
10

C:\prakashclasses>java Test
Enter n value:
12345
15
```

08.Implement a program to calcualte reverse of the given number?

~~~~~~

LOL- Lots Of Love for math

formula:

```
            ((n%10)*pow(10,len-1))+rev(n/10,--len)

n=123,len=3 ----> 3*pow(10,2) + rev(12,2) ---> 3*100=300
n=12,len=2 -----> 2*pow(10,1) + rev(1,1) ----> 2*10=20
n=1,len=1 ------> 1*pow(10,0) + rev(0,0) ----> 1*1=1
n=0 ------------> 0

300+20+1+0 ===> 321
```

formula:

```
            ((n%10)*pow(10,len-1))+rev(n/10,--len)

n=98123, len=5 ----> 3*pow(10,4) + rev(9812,4) ----> 3*10000=30000
n=9812, len=4 -----> 2*pow(10,3) + rev(981,3) -----> 2*1000 = 2000
n=981, len=3 ------> 1*pow(10,2) + rev(98,2) ------> 1*100  =  100
n=98, len=2 -------> 8*pow(10,1) + rev(9,1) -------> 8*10   =   80
n=9, len=1 --------> 9*pow(10,0) + rev(0,0) -------> 9*1    =    9
```
~~~~~~

```
n=0 ---------------> terminate -------------------->         32189
```

rev(98123) = 32189

Ex:

```
import java.util.*;

class Demo
{
	static int reverse(int n,int len)
	{
		if(n==0)
			return 0;
		else
			return ((n%10)*(int)Math.pow(10,len-1)) + reverse(n/10,-
-len);

	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		String s = obj.nextLine();

	System.out.println(Demo.reverse(Integer.parseInt(s),s.length()));//
reverse of 'n'
	}
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
781902
209187

C:\prakashclasses>java Test
Ten
Exception in thread "main"
java.lang.NumberFormatException: For input string: "Ten"
at java.base/java.lang.NumberFormatException.forInputString
(NumberFormatException.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:668)
        at java.base/java.lang.Integer.parseInt(Integer.java:784)
        at Test.main(Test.java:21)
```

09. Implement a program to count number of digits present in the given number

```
--------------------------------------------------------------------------
----
import java.util.*;

class Demo
{
      static int c=0;
      static int count(int n)
      {
            if(n!=0)
            {
                  c++;
                  count(n/10);
            }
            return (c!=0)?c:1;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int n = obj.nextInt();
            System.out.println(Demo.count(n));
      }
}
```

output:

```
C:\prakashclasses>java Test
12345
5

C:\prakashclasses>java Test
123
3

C:\prakashclasses>java Test
12
2

C:\prakashclasses>java Test
1
1

C:\prakashclasses>java Test
0
1
```

Note:

Octal Constants

C ----> prefixed with 0

```
            C++ --> prefixed with 0
            Java -> prefixed with 0
            Py ---> prefixed with 0o or 0O
```

10. Implement a program to convert decimal number into binary?
~~~~~~

```java
import java.util.*;

class Demo
{
      static int convert(int n)
      {
            if(n==0)
                  return 0;
            else
                  return (n%2+10*convert(n/2));
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int n = obj.nextInt();
            System.out.println(Demo.convert(n));
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
8
1000

C:\prakashclasses>java Test
10
1010

C:\prakashclasses>java Test
19
10011

C:\prakashclasses>java Test
556
1000101100
```

Ex:
n=12

```
1+10*0 = 1
1+10*1 = 11
```
~~~~~~

```
0+10*11 = 110
0+10*110 = 1100
```

version2:

```
import java.util.*;

class Demo
{
      static void convert(int n)
      {
            if(n==0)
            {
                  System.out.print("");
            }
            else
            {
                  convert(n/2);
                  System.out.print(n%2);
            }
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int n = obj.nextInt();
            Demo.convert(n);
      }
}
```

Logic:

```
      static void convert(int n)
      {
            if(n!=0)
            {
                  convert(n/2);
                  System.out.print(n%2);
            }
      }
```

11. Implement a program to find nth fib number.

0, 1, 1, 2, 3, 5, and so on

```
f(n)=n  if n=0 or n=1
f(n)=f(n-1)+f(n-2)  if n>1


f(0) = 0
```

```
f(1) = 1
f(2) = f(2-1)+f(2-2) = f(1) + f(0) = 1 + 0 = 1
f(3) = f(3-1)+f(3-2) = f(2) + f(1) = 1 + 1 = 2
f(4) = f(4-1)+f(4-2) = f(3) + f(2) = 2 + 1 = 3
```

and so on

```
f(n) = f(n-1) + f(n-2)
```

Ex:

```java
import java.util.*;

class Demo
{
	static int fib(int n)
	{
		if(n==0 || n==1)
			return n;
		else
			return fib(n-1)+fib(n-2);
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		int n = obj.nextInt();
		for(int i=0;i<n;i++){
			System.out.print(Demo.fib(i)+", ");
		}
	}
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
5
0, 1, 1, 2, 3,
C:\prakashclasses>java Test
10
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

12) Implement a program to find LCM of two numbers?

Least/Lowest Common Multiple

```java
import java.util.*;
```

```
class Demo
{
      static int com=1;
      static int lcm(int n1,int n2)
      {
            if(com%n1==0 && com%n2==0)
                  return com;
            com++;
            lcm(n1,n2);
            return com;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter n1 value:");
            int n1=obj.nextInt();
            System.out.println("Enter n2 value:");
            int n2=obj.nextInt();
            System.out.println(Demo.lcm(n1,n2));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n1 value:
4
Enter n2 value:
6
12

C:\prakashclasses>java Test
Enter n1 value:
6
Enter n2 value:
9
18

C:\prakashclasses>java Test
Enter n1 value:
5
Enter n2 value:
10
10
```

13) Implement a program to find HCF/GCD of the given two numbers

```
-----------------------------------------------------------------
```

Highest Common Factors
Greatest Common Divisior

```
import java.util.*;

class Demo
{
	static int gcd(int a,int b)
	{
		while(a!=b)
		{
			if(a>b)
				return gcd(a-b,b);
			else
				return gcd(a,b-a);
		}
		return a;
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		System.out.println("Enter n1 value:");
		int n1=obj.nextInt();
		System.out.println("Enter n2 value:");
		int n2=obj.nextInt();
		System.out.println(Demo.gcd(n1,n2));
	}
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter n1 value:
4
Enter n2 value:
6
2
```

14) Implement a program to find reverse of the given string using recursion?

```
---------------------------------------------------------------------------
---
import java.util.*;

class Demo
{
	static String strrev(String s)
```

```
    {
        if(s==null || s.length()<=1)//BC
            return s;
        return strrev(s.substring(1))+s.charAt(0);
    }
}

class Test
{
    public static void main(String[] args)
    {
        Scanner obj = new Scanner(System.in);
        System.out.println("Enter any String:");
        String s = obj.nextLine();
        System.out.println(Demo.strrev(s));
    }
}

ouput:
------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any String:
abc
cba

15) Implement a program to remove the given character from a string?
--------------------------------------------------------------------
"abcde"     ----> a|b|c|d|e
"axbxcxdxex" ---> a|x|b|x|c|x|d|x|e|x

x
if ch is not x then
return strrev(s.substring(1))+s.charAt(0);
else
return strrev(s.substring(1));

"axbxcxdxex" ---> a|b|c|d|e
L to R       ---> R to L

"axbxcxdxex" ---> e|d|c|b|a
R to L  --------> R to L

abcde

import java.util.*;

class Demo
{
    static String nox(String s,int index)
    {
        if(index<0) //Base Condition
            return "";
            if(s.charAt(index)=='x') //RC1: if char is 'x', i.e. remove
                  return nox(s,index-1); //Recursion, ignoring char
            else //RC2: if char is not 'x', i.e. it should be stored
                  return nox(s,index-1)+s.charAt(index); //recursion,
store char in stack
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter any string:");
            String s = obj.nextLine();
            System.out.println(Demo.nox(s,s.length()-1));//axbx,4-1=3
      }
}
```

```
output:
-------
C:\prakashclasses>java Test
Enter any string:
abcd
abcd

C:\prakashclasses>java Test
Enter any string:
axbcd
abcd

C:\prakashclasses>java Test
Enter any string:
axbxcd
abcd

C:\prakashclasses>java Test
Enter any string:
axbxcxdx
abcd
```

16) Implement a program to return a new String, where all the adjacent characters are seperated by a "*".

```
--------------------------------------------------------------------------
------------
"hello" ----> "h*e*l*l*o"
"abc" ------> "a*b*c"
"ab" -------> "a*b"

      static String newS(String s,int index)
      {
            if(index<1)
                  return s.charAt(index);
            return newS(s,index-1)+"*"+s.charAt(index);
      }
```

Ex:

```
import java.util.*;

class Demo
{
      static String newS(String s,int index)
      {
            if(index<1)
                  return s.substring(0,index+1);//s.charAt(index)+"";
            return newS(s,index-1)+"*"+s.charAt(index);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter any string:");
            String s = obj.nextLine();
            System.out.println(Demo.newS(s,s.length()-1));//abc --->
a*b*c
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
abcdef
a*b*c*d*e*f
```

17) Implement a program to return new string where identical adjcent chars are sep by *

Ex:

```
abc ----> abc
hello --> hel*lo
xxyy ---> x*xy*y
```

```
      static String newS(String s,int index)
      {
            if(index<1)
                  return s.substring(0,index+1);
            if(s.charAt(index-1)==s.charAt(index))
                  return newS(s,index-1)+"*"+s.charAt(index);
            else
                  return newS(s,index-1)+s.charAt(index);
      }

import java.util.*;

class Demo
{
      static String newS(String s,int index)
      {
            if(index<1)
                  return s.substring(0,index+1);
            if(s.charAt(index-1)==s.charAt(index))
                  return newS(s,index-1)+"*"+s.charAt(index);
            else
                  return newS(s,index-1)+s.charAt(index);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter any string:");
            String s = obj.nextLine();
            System.out.println(Demo.newS(s,s.length()-1));//abc --->
a*b*c
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
abc
abc

C:\prakashclasses>java Test
Enter any string:
abbc
ab*bc

C:\prakashclasses>java Test
Enter any string:
hello
hel*lo

C:\prakashclasses>java Test
Enter any string:
aabbcc
a*ab*bc*c
```

18) Implement a program to return true if a string nesting of zero or more pairs of ()

```
--------------------------------------------------------------------------
-------------

"()" ----> true
"(())" --> true
"(((" --> false
```

```
import java.util.*;

class Demo
{
      static boolean newS(String s,int i,int j)
      {
            if(i>j)
                  return true;
            if(s.charAt(i)=='(' && s.charAt(j)==')')
                  return newS(s,i+1,j-1);
            else
                  return false;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter any string:");
            String s = obj.nextLine();
            System.out.println(Demo.newS(s,0,s.length()-1));
      }
}
```

```
output:
-------
C:\prakashclasses>java Test
Enter any string:
()
true

C:\prakashclasses>java Test
Enter any string:
((
false

C:\prakashclasses>java Test
Enter any string:
(())
true
```

```
C:\prakashclasses>java Test
Enter any string:
((a))
false
```

18) Implement a program to count number of times, the give char occurred.

```
--------------------------------------------------------------------------
import java.util.*;

class Demo
{
      static int count(String s,char ch,int index) //x
      {
            if(index<0)
                  return 0;
            //if(s.charAt(index)=='x')

      //if(s.charAt(index)=='a'||s.charAt(index)=='e'||s.charAt(index)=='
i'||s.charAt(index)=='o'||s.charAt(index)=='u')
            if(s.charAt(index)==ch)
                  return 1+count(s,ch,index-1);
            else
                  return count(s,ch,index-1);
      }
}


class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter any string:");
            String s = obj.nextLine();
            System.out.println(Demo.count(s,'a',s.length()-1));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
prakash
2

C:\prakashclasses>java Test
Enter any string:
welcome
0
```

Note: compared to loops recursion is easy sir iff base condition is very strong

19) IMP to replace the given old character with new character in the original string?

```
--------------------------------------------------------------------------
------------
'x' --------> 'y'
"codex" ----> "codey"
"xxhixx" ---> "yyhiyy"
"xbix" -----> "ybiy"
```

```
import java.util.*;

class Demo
{
	static String replace(String s,int index)
	{
		//Base condition
		if(index<0)
			return "";
		if(s.charAt(index)=='x')
			return replace(s,index-1)+"y";
		else
			return replace(s,index-1)+s.charAt(index);
	}
}


class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		System.out.println("Enter any string:");
		String s = obj.nextLine();
		System.out.println(Demo.replace(s,s.length()-1));
	}
}
```

output:

```
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
codex
codey

C:\prakashclasses>java Test
Enter any string:
xhix
yhiy
```

```
C:\prakashclasses>java Test
Enter any string:
xxByexx
yyByeyy
```

replace()
loop & stringbuffer/builder
recursion

20) IMP to count the number of times given string appeared in the original string?


```
"python is very very easy programming" ----> 2
"java is very easy" -----------------------> 1
"c programming is easy" -------------------> 0
```

```
import java.util.*;

class Demo
{
      static int count(String s,int index)
      {
            //base condition
            if(index<3)
                  return 0;
            if(s.substring(index-3,index+1).equals("very")) //RC1==> if
'very' word is existed
                  return 1+count(s,index-3);
            else ////RC2==> if 'very' word is not existed
                  return count(s,index-1);
      }
}


class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter any string:");
            String s = obj.nextLine();//very,3
            System.out.println(Demo.count(s,s.length()-1));
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
c programming is easy
```

```
0

C:\prakashclasses>java Test
Enter any string:
java is very easy
1

C:\prakashclasses>java Test
Enter any string:
python is very very easy
2

C:\prakashclasses>java Test
Enter any string:
he is very good and descent
1

C:\prakashclasses>java Test
Enter any string:
 veryvery a cvery
3

C:\prakashclasses>java Test
Enter any string:
veryabcdvery
2
```

21) IMP to replace the given string with new string?

```
==
.equals()

xpix ----> x3.147x
pip -----> 3.147p
abc -----> abc
ab ------> ab
a -------> a

"pi" replaced with "3.147"

import java.util.*;

class Demo
{
	static String replacestr(String s,int index)
	{
		//base condition Eg:a
		if(index<1)
			return s.substring(0,index+1);//to return original str
		if(s.substring(index-1,index+1).equals("pi")) //RC1==> if
'pi' word is existed
			return replacestr(s,index-2)+"3.147";
		else ////RC2==> if 'pi' word is not existed, Eg: pix
			return replacestr(s,index-1)+s.charAt(index);
	}
}


class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		System.out.println("Enter any string:");
		String s = obj.nextLine();//very,3
		System.out.println(Demo.replacestr(s,s.length()-1));
	}
}
```

output:

```
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
xpix
x3.147x

C:\prakashclasses>java Test
Enter any string:
xpxx
xpxx

C:\prakashclasses>java Test
Enter any string:
xpjx
xpjx

C:\prakashclasses>java Test
Enter any string:
pi
3.147

C:\prakashclasses>java Test
Enter any string:
abc
abc
```

Towers of Hanoi

It is a problem, where we have to move the disks from source to destination. by following the rules

R1----> at a time only one disk we have to move
R2----> place smaller disk on the top of larger disk

```
Ex:
---
import java.util.*;

class Demo
{
      static void towersOfHanoi(int n,String src,String helper,String
dest)
      {
            if(n==1){
                  System.out.println("Move The Disk "+n+" from "+src+" to
"+dest);
                  return;
            }
            towersOfHanoi(n-1,src,dest,helper);
            System.out.println("Move The Disk "+n+" from "+src+" to
"+dest);
            towersOfHanoi(n-1,helper,src,dest);
      }
}


class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter number of disks:");
            int n=obj.nextInt();

            Demo.towersOfHanoi(n,"S","H","D");
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of disks:
1
Move The Disk 1 from S to D


C:\prakashclasses>java Test
Enter number of disks:
2
Move The Disk 1 from S to H
Move The Disk 2 from S to D
Move The Disk 1 from H to D

C:\prakashclasses>java Test
Enter number of disks:
```

```
3
Move The Disk 1 from S to D
Move The Disk 2 from S to H
Move The Disk 1 from D to H
Move The Disk 3 from S to D
Move The Disk 1 from H to S
Move The Disk 2 from H to D
Move The Disk 1 from S to D

C:\prakashclasses>java Test
Enter number of disks:
4
Move The Disk 1 from S to H
Move The Disk 2 from S to D
Move The Disk 1 from H to D
Move The Disk 3 from S to H
Move The Disk 1 from D to S
Move The Disk 2 from D to H
Move The Disk 1 from S to H
Move The Disk 4 from S to D
Move The Disk 1 from H to D
Move The Disk 2 from H to S
Move The Disk 1 from D to S
Move The Disk 3 from H to D
Move The Disk 1 from S to H
Move The Disk 2 from S to D
Move The Disk 1 from H to D

C:\prakashclasses>java Test
Enter number of disks:
5
Move The Disk 1 from S to D
Move The Disk 2 from S to H
Move The Disk 1 from D to H
Move The Disk 3 from S to D
Move The Disk 1 from H to S
Move The Disk 2 from H to D
Move The Disk 1 from S to D
Move The Disk 4 from S to H
Move The Disk 1 from D to H
Move The Disk 2 from D to S
Move The Disk 1 from H to S
Move The Disk 3 from D to H
Move The Disk 1 from S to D
Move The Disk 2 from S to H
Move The Disk 1 from D to H
Move The Disk 5 from S to D
Move The Disk 1 from H to S
Move The Disk 2 from H to D
Move The Disk 1 from S to D
Move The Disk 3 from H to S
Move The Disk 1 from D to H
Move The Disk 2 from D to S
Move The Disk 1 from H to S
Move The Disk 4 from H to D
Move The Disk 1 from S to D
Move The Disk 2 from S to H
Move The Disk 1 from D to H
Move The Disk 3 from S to D
Move The Disk 1 from H to S
Move The Disk 2 from H to D
Move The Disk 1 from S to D
```

1) write recursive method
2) n=1 new paper
3) n=2 new paper
4) n=3 new paper
5) n=4 new paper
6) n=5 new paper

Arrays:

introduction to array:

==> a variable can hold only one value or item at a time.

```
int x = 10;
System.out.println(x); -----> 10
x=999;
System.out.println(x); -----> 999
```

==> can hold only one value at a time.

==> store all 50 students id numbers.

```
int s1;
int s1,s2;
int s1,s2,s3;
int s1,s2,s3,s4;
int s1,s2,s3,s4,......,s50;
```

==> s1+s2+s3+............+s50

==> array is a collection or group of items (values).

==> all these values are stored under a common name.

==> all these items must be of same type.

==> it is accesses/represented by using 'index'

Ex:

```
     int a1,a2,a3,a4,a5,a6,a7,a8,a9,a10;

     int a[10];
```

```
      a[0], a[1], a[2], a[3], .... a[n-1]

      where n is size of the array, n=10

      index ---> 0 to 9

==> array is considered as a data structure.

advantages and disadvantages of arrays:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
advantages:
-----------
                 1. it collects group of items.
                 2. only one name is sufficient to represent all the
values.
                 3. we have index support is there.
                 4. insertion order is preserved.
                 5. duplicate elements|items allowed.

disadvantages:
--------------
                 1. it is fixed in size. [not growable]
                 2. it collects only same type of elements [homogeneous]
                 3. no built methods.

variable ----> arrays ----> collections
variable ----> arrays ----> ADTs [Abstract Data Type]

struct Stack
{
      --------
      --------
}

Internet Centres -----> Mobile, Laptop, Tabs, ....



declaration of an array:
~~~~~~~~~~~~~~~~~~~~~~~~
Once if we are using any variable or an array, in java, first we have to
declare it. The following are the various declarations supported by java.

syntax:
           datatype arrayname[]; ---------------> 1-D
           datatype arrayname[][]; -------------> 2-D
           datatype arrayname[][][]; -----------> 3-D
           datatype arrayname[][]....[]; -------> multi-D or n-D

Ex:
           int a[];
           int []a;
           int[] a; ----> recommended
```

Ex:

```
            int a[][];
            int [][]a;
            int[][] a;
            int[] []a;
            int[] a[];
            int []a[];
```

Note: At the time of declaration of an array, we should provide size, if we provide size, then we will get error.

Ex:

```
            int a[]; ----> valid in java
            int a[10]; --> invalid in java

            int a[]; ----> invalid in c
            int a[10]; --> valid in c
```

Note: Java internally is providing any support for arrays, they have defined inbuilt ADTs

```
class I[
{

}

class F[
{

}

class Test
{
      public static void main(String[] args)
      {
            int a[][]=new int[3][3];
            System.out.println(a);
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
[I@76ed5528

C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
[[I@76ed5528
```

creation of an array:
~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~

Once if declaration got completed, we have allocate memory for an array, because java arrays are considered as objects, we can create array object by using 'new' keyword with the following syntax.

```
syntax:
            datatype arrayname[];
            arrayname = new datatype[size];

Ex:
            int a[];
            a = new int[3];

Ex:
            String names[];
            names = String[10];

Ex:
            Emp e[];
            e = new Emp[100];

Ex:
            int a[][];
            a=new int[3][3];
```

Note: multi-D array are not represented in matrix style. It is represented by using 'array of arrays'

```
intialization of arrays:
~~~~~~~~~~~~~~~~~~~~~~~~
```

=> In c,c++ programming, default value concept is not existed, once if we create an array, it hold garbage values, but in java, once if an array is created, if we are not providing any value, it takes default values.

```
primitive data types ---> 0, 0.0, '', false
objects ----------------> null

Ex:
            int a[];
            a=new int[3];

            System.out.println(a[0]); ---> 0
            System.out.println(a[1]); ---> 0
            System.out.println(a[2]); ---> 0
Ex:
            boolean a[];
            a=new boolean[3];

            System.out.println(a[0]); ---> false
            System.out.println(a[1]); ---> false
            System.out.println(a[2]); ---> false

Ex:
            String a[];
```

```
		a=new String[3];

		System.out.println(a[0]); ---> null
		System.out.println(a[1]); ---> null
		System.out.println(a[2]); ---> null
```

declare, create and intialization in a single line:
~~~~~~
If we know the array elements in advance, then we can declare, create and we can perform intializations in a single line as follows.

Ex:

```
		int a[] = {1,2,3,4,5};
		System.out.println(a[0]);//1
		System.out.println(a[1]);//2
		System.out.println(a[2]);//3
		System.out.println(a[3]);//4
		System.out.println(a[4]);//5

		int a[][] = {{1,2,3},{4,5,6},{7,8,9}};

		System.out.println(a[0][0]);//1
		System.out.println(a[0][1]);//2
		System.out.println(a[0][2]);//3

		System.out.println(a[1][0]);//4
		System.out.println(a[1][1]);//5
		System.out.println(a[1][2]);//6

		System.out.println(a[2][0]);//7
		System.out.println(a[2][1]);//8
		System.out.println(a[2][2]);//9
```

length identifier:
~~~~~~~~~~~~~~~~~~
if we dn't know the number of elements in an array, then we can use 'length' identifier to find the number of elements in an array.

Ex:

```
		int[] a = {11,222,444,222,333};

		System.out.println(a.length);//----> 5

		int a[][] = {{1,2,3},{4,5,6},{7,8,9}};

		System.out.println(a.length);//Error

		System.out.println(a[0].length);//3
		System.out.println(a[1].length);//3
		System.out.println(a[2].length);//3
```

index concept:

~~~~~~~~~~~~~~

==> we can traverse or retrive or access array elements by using 'index' concept.
==> index is always an integer value.
==> it must be always +ve integer values.
==> we have to enclose this index value inside subscripts[].

Ex:

```
class Test
{
      public static void main(String[] args)
      {
            int a[] = new int[3];

            System.out.println(a.length);//3

            System.out.println(a[0]);//0
            System.out.println(a[1]);//0
            System.out.println(a[2]);//0

            a[0] = 444;
            a[1] = 555;
            a[2] = 666;

            System.out.println(a[0]);//444
            System.out.println(a[1]);//555
            System.out.println(a[2]);//666
      }
}
```

output:

```
3
0
0
0
444
555
666
```

ArrayIndexOutOfBoundsException:-

When we are trying to access the elements which are not in our array range, then java raises automatically runtime error saying "ArrayIndexOutOfBoundsException".

Ex:

```
class Test
{
      public static void main(String[] args)
      {
            int a[] = new int[3];
```
~~~~~~~~~~~~~~

```
            System.out.println(a.length);//3

            System.out.println(a[0]);//0
            System.out.println(a[1]);//0
            System.out.println(a[2]);//0
            System.out.println(a[3]);//RE:
      }
}
```

output:

```
3
0
0
0
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException:
Index 3 out of bounds for length 3
        at Test.main(Test.java:13)
```

NegativeArraySizeException:

When we are creating an array, if we are using -ve integer value for array size, then java raises automatically runtime error saying "NegativeArraySizeException".

Ex:

```
class Test
{
      public static void main(String[] args)
      {
            int a[] = new int[-3];
      }
}
```

output:

```
Exception in thread "main" java.lang.NegativeArraySizeException: -3
        at Test.main(Test.java:6)
```

Reading and Writing Array Elements:

Reading elements from user

```
Scanner obj = new Scanner(System.in);

            obj.nextByte();
            obj.nextShort();
            obj.nextInt();
            obj.nextLong();
            obj.nextFloat();
            obj.nextDouble();
            obj.nextBoolean();
            obj.next() or obj.nextLine()

            next()     ---> stop reading data when a space is encountered
            nextLine() ---> stop reading data when a new line is encountered
```

Writing elements to user

1. index concept
2. while loop
3. for loop *
4. for each loop **

array ---> hard code
collection --> ready made methods ---> index concept

sir while assigning 2d array ex.
int[][] x = new int[][4]; getting an error

```
int[][] x = new int[3][];

x[0] = new int[3];
x[1] = new int[4];
x[2] = new int[1];
```

Ex:

```
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            System.out.println("Array elements by using while loop..");
            int index=0;
            while(index<a.length)
            {
                  System.out.println("index="+index+"
a["+index+"]="+a[index]);
                  index++;
            }
```

```
        System.out.println("Array elements by using for loop..");
        for(i=0;i<a.length;i++)
        {
            System.out.println("index="+i+" a["+i+"]="+a[i]);
        }

        System.out.println("Array elements by using for each 
loop..");
        for(int item:a)
        {
            System.out.println("item="+item);
        }
    }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
999
123
555
222
999
Array elements by using while loop..
index=0 a[0]=999
index=1 a[1]=123
index=2 a[2]=555
index=3 a[3]=222
index=4 a[4]=999
Array elements by using for loop..
index=0 a[0]=999
index=1 a[1]=123
index=2 a[2]=555
index=3 a[3]=222
index=4 a[4]=999
Array elements by using for each loop..
item=999
item=123
item=555
item=222
item=999
```

```
sum of elements in array
------------------------
==> using for loop
==> using for each loop
```

Ex:

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

```
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int sum=0,i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            sum=0;
            for(i=0;i<a.length;i++)
            {
                  sum=sum+a[i];
            }
            System.out.println(sum);

      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
12
10
3
11
13
49
```

Program to read and write array elements.

Program to calcualte sum of elements present in an array.

Program to calcualte sum of even elements present in an array.

Program to calcualte sum of odd elements present in an array.

Program to calcualte sum of +ve elements present in an array.

Program to calcualte sum of -ve elements present in an array.

Program to calcualte sum of elements which are divisible by 2 and 3 present in an array.

Program to calcualte sum of prime elements present in an array.

Program to calcualte sum of all elements factorials present in an array.

Logic:

```
            sum = 0;
            for(i=0;i<a.length;i++)
            {
                  if(-----){
                        sum=sum+a[i];
                  }
            }
            s.o.p(sum);
```

iseven(a[i]),isodd,isprime,ispve,isnve,sum,etc

Program to find max and min element present in an array

```
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int max,min,i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            max=a[0];
            for(i=1;i<a.length;i++)
            {
                  if(max<a[i])
                  {
                        max=a[i];
                  }
            }
            System.out.println("max="+max);

            min=a[0];
            for(i=1;i<a.length;i++)
            {
                  if(min>a[i])
                  {
                        min=a[i];
                  }
            }
            System.out.println("min="+min);
      }
```

```
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
3
5
2
4
1
max=5
min=1


Program to replace old element with new element
-----------------------------------------------
version1: update all occurrences
--------------------------------
Ex:
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            System.out.println("Array Elements Before update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            //logic
            int olde,newe;
            System.out.println("Enter old element");
            olde=obj.nextInt();
            System.out.println("Enter new element");
            newe=obj.nextInt();
            for(i=0;i<a.length;i++)
            {
                  if(olde==a[i]){
                        a[i]=newe;
                  }
            }

            System.out.println("Array Elements After update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");

      }
}
```

output:

```
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
1 2 1 2 3
Array Elements Before update...
1 2 1 2 3
Enter old element
2
Enter new element
9
Array Elements After update...
1 9 1 9 3
```

version2: update only first occurrence

```
--------------------------------------
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            System.out.println("Array Elements Before update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            //logic
            int olde,newe;
            System.out.println("Enter old element");
            olde=obj.nextInt();
            System.out.println("Enter new element");
            newe=obj.nextInt();
            for(i=0;i<a.length;i++)
            {
                  if(olde==a[i]){
                        a[i]=newe;
                        break;
                  }
            }

            System.out.println("Array Elements After update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");

      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
1 2 3 1 2
Array Elements Before update...
1 2 3 1 2
Enter old element
2
Enter new element
8
Array Elements After update...
1 8 3 1 2
```

```
version3: update only second occurrence
---------------------------------------
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();
            int i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            System.out.println("Array Elements Before update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            //logic
            int olde,newe;
            System.out.println("Enter old element");
            olde=obj.nextInt();
            System.out.println("Enter new element");
            newe=obj.nextInt();
            int c=0;
            for(i=0;i<a.length;i++)
            {
                  if(olde==a[i]){
                        c++;
                        if(c==2)
                        {
                              a[i]=newe;
                              break;
                        }
                  }
            }

            System.out.println("Array Elements After update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");

      }
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
7
Enter 7 elements...
1 2 3 1 2 1 2
Array Elements Before update...
1 2 3 1 2 1 2
Enter old element
2
Enter new element
44
Array Elements After update...
```

```
1 2 3 1 44 1 2
```

version4: last occurrence

```
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            System.out.println("Array Elements Before update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            //logic
            int olde,newe;
            System.out.println("Enter old element");
            olde=obj.nextInt();
            System.out.println("Enter new element");
            newe=obj.nextInt();
            int c=0;
            for(i=a.length-1;i>=0;i--)
            {
                  if(olde==a[i]){
                              a[i]=newe;
                              break;
                  }
            }

            System.out.println("Array Elements After update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");

      }
}
```

output:

```
C:\prakashclasses>javac Test.java
```

```
C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
1 2 3 1 2 5
Array Elements Before update...
1 2 3 1 2 5
Enter old element
2
Enter new element
7
Array Elements After update...
1 2 3 1 7 5
```

Program to replace given location with new element

```
import java.util.Scanner;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int size = obj.nextInt();

            int i,a[] = new int[size];

            System.out.println("Enter "+size+" elements...");
            for(i=0;i<a.length;i++){
                  a[i] = obj.nextInt();
            }

            System.out.println("Array Elements Before update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            //logic
            int index,newe;
            System.out.println("Enter index value:");
            index=obj.nextInt();
            if(index>=0 && index<a.length){
                  System.out.println("Enter new element");
                  newe=obj.nextInt();
                  a[index]=newe;
            }
            else{
                  System.out.println("ArrayIndexOutOfBoundsException");
            }

            System.out.println("Array Elements After update...");
            for(i=0;i<a.length;i++)
                  System.out.print(a[i]+" ");

      }
}
```

```
output:
--------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
10 20 30 40 50 60
Array Elements Before update...
10 20 30 40 50 60
Enter index value:
2
Enter new element
90
Array Elements After update...
10 20 90 40 50 60
C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
10 20 30 40 50 60
Array Elements Before update...
10 20 30 40 50 60
Enter index value:
9
ArrayIndexOutOfBoundsException
Array Elements After update...
10 20 30 40 50 60
```

Program to sort the elements present in the array

version1: sort the elements in asc order

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,j,t,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements.");
            for(i=0;i<n;i++)
```

```
                  a[i] = obj.nextInt();

            System.out.println("Array Elements Before Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

            //version1==> sort the data in asc order
            for(i=0;i<n;i++)
            {
                  for(j=i+1;j<n;j++)
                  {
                        if(a[i]>a[j]){
                              t = a[i];
                              a[i]=a[j];
                              a[j]=t;
                        }
                  }
            }

            System.out.println();
            System.out.println("Array Elements After Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements.
1 3 2 5 4
Array Elements Before Sorting:
1 3 2 5 4
Array Elements After Sorting:
1 2 3 4 5
```

```
String s = "acbed";
s.toCharArray() ----> {'a','c','b','e','d'} ---> {'a''b','c','e','d'}
```

```
version2: sort the elements in desc order
----------------------------------------
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,j,t,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements.");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            System.out.println("Array Elements Before Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

            //version2==> sort the data in desc order
            for(i=0;i<n;i++)
            {
                  for(j=i+1;j<n;j++)
                  {
                        if(a[i]<a[j]){
                              t = a[i];
                              a[i]=a[j];
                              a[j]=t;
                        }
                  }
            }

            System.out.println();
            System.out.println("Array Elements After Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements.
1 4 2 5 3
Array Elements Before Sorting:
1 4 2 5 3
Array Elements After Sorting:
5 4 3 2 1
```

version3: sorting an array by using predefined classes

Arrays.sort(a)

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements.");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            System.out.println("Array Elements Before Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

            //version3==> by using predefined methods ASC
            Arrays.sort(a);

            System.out.println();
            System.out.println("Array Elements After Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements.
1 4 2 5 6 3
Array Elements Before Sorting:
1 4 2 5 6 3
Array Elements After Sorting:
1 2 3 4 5 6
```

version4: by using predefined only but desc

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements.");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            System.out.println("Array Elements Before Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

            //version4==> by using predefined methods ASC
            Arrays.sort(a);

            System.out.println();
            System.out.println("Array Elements After Sorting:");
            for(i=n-1;i>=0;i--)
                  System.out.print(a[i]+" ");

      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements.
1 4 2 5 6 3
Array Elements Before Sorting:
1 4 2 5 6 3
Array Elements After Sorting:
6 5 4 3 2 1
```

version5: customized sorting

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements.");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            System.out.println("Array Elements Before Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");
            //version5==> by using customized sorting
            //start and ending location
            Arrays.sort(a,0,n/2);

            System.out.println();
            System.out.println("Array Elements After Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

      }
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements.
1 6 2 5 4 3
Array Elements Before Sorting:
1 6 2 5 4 3
Array Elements After Sorting:
1 2 6 5 4 3

Ex:
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements.");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            System.out.println("Array Elements Before Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");

            //version5==> by using customized sorting
            //start and ending location
            Arrays.sort(a,n/2,n);

            System.out.println();
            System.out.println("Array Elements After Sorting:");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");
      }
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements.
1 6 2 5 4 3
Array Elements Before Sorting:
1 6 2 5 4 3
Array Elements After Sorting:
1 6 2 3 4 5

sort asc ----> 0 to n/2, n to n/2

for(i=0;i<n/2;i++){}
for(i=n/2;i<n;i++){}

Ex:
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter any string:");
            String s = obj.nextLine();

            char t,a[] = s.toCharArray();
            //Arrays.sort(ch);
            for(int i=0;i<a.length;i++)
            {
                  for(int j=i+1;j<a.length;j++)
                  {
                        if(a[i]>a[j]){
                              t = a[i];
                              a[i]=a[j];
                              a[j]=t;
                        }
                  }
            }

            System.out.println(new String(a));
      }
}

output:
------
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter any string:
azbc1m3d2
123abcdmz
```

```
After sorting the elements ---> min.........max

a[0],a[1],a[2],........a[n-2],a[n-1]


1st min -----> a[1-1] ----> a[0]
2nd min -----> a[2-1] ----> a[1]
3rd min -----> a[3-1] ----> a[2]
.
.
nth min -----> a[n-1]

Ex:
      5
      4 1 3 5 2
      1 2 3 4 5

1st min ---> a[1-1] = a[0] = 1
2nd min ---> a[2-1] = a[1] = 2
3rd min ---> a[3-1] = a[2] = 3
4th min ---> a[4-1] = a[3] = 4
5th min ---> a[5-1] = a[4] = 5


a[0],a[1],a[2],........a[n-2],a[n-1]

1st max -----> a[n-1]
2nd max -----> a[n-2]
3rd max -----> a[n-3]
4th max -----> a[n-4]
5th max -----> a[n-5]
.
.
nth max -----> a[n-n] = a[0]


Ex:
      5
      4 1 3 5 2
      1 2 3 4 5

1st max ----> a[5-1] = a[4] = 5
2nd max ----> a[5-2] = a[3] = 4
3rd max ----> a[5-3] = a[2] = 3
4th max ----> a[5-4] = a[1] = 2
```

5th max ----> a[5-5] = a[0] = 1

Program to find max and min element present in an array.
Program to find 2nd max and 2nd min element present in an array.
Program to find 3rd max and 3rd min element present in an array.

kprakashbabutechnicaltrainer@gmail.com

Program to print 1st smallest, 1st largest, 2nd smallest, 2nd largest and so on

```
--------------------------------------------------------------------------
----------
n=5
1 4 2 5 3

1 2 3 4 5

output: 1 5 2 3 4 4

n=6
1 4 6 2 5 3

1 2 3 4 5 6

output: 1 6 2 5 3 4
```

```
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements...");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            Arrays.sort(a);

            int low,high;
            low = 0;
            high = n-1;
            while(low<=high)
            {
                  System.out.print(a[low]+" "+a[high]+" ");
                  low++;
                  high--;
            }
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
1 5 3 2 4
1 5 2 4 3 3
C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
1 5 3 6 4 2
1 6 2 5 3 4
```

```
Print all elements in an array in wave form
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ex:
      n = 6
      1 5 2 4 6 3

      1 2 3 4 5 6

output:

            1 3 2 5 4 6
```

```
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int t,i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements...");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            Arrays.sort(a);

            System.out.print(a[0]+" ");
```

```
            for(i=1;i<n-1;i=i+2)
            {
                  t=a[i];
                  a[i]=a[i+1];
                  a[i+1]=t;
                  System.out.print(a[i]+" "+a[i+1]+" ");
            }
            if(n%2==0)
                  System.out.print(a[i]);
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
1 5 3 4 6 2
1 3 2 5 4 6
C:\prakashclasses>java Test
Enter array size:
5
Enter 5 elements...
1 5 2 3 4
1 3 2 5 4
```

searching:
----

linear search and binary search

Implement a program to search for an element in an array

Ex:

```
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int t,i,n = obj.nextInt();

            int a[] = new int[n];
            System.out.println("Enter "+n+" elements...");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();
```

```
        System.out.println("Enter the element to search:");
        int key = obj.nextInt();

        int index = -1;

        for(i=0;i<n;i++)
        {
            if(key==a[i]){
                index = i;
                break;
            }
        }

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

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
1 5 2 3 4 7
Enter the element to search:
5
1

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 elements...
1 5 2 3 4 7
Enter the element to search:
9
-1
```

sir array elements are 1 2 3 5 5 how to find second largest

```
1 2 3 5 5
0 1 2 3 4

2nd max -----> a[5-2] ---> a[3] = 5
```

```
binary search:
~~~~~~~~~~~~~~
```

Here first we have to sort the elements in asc order, then compare key element with middle elements if result found then return the result else we can compare either in left part or right part.

Ex:

```
import java.util.*;

class Demo
{
	static int binarysearch(int a[],int key){
		int l=0,h=a.length-1,mid;

		while(l<=h){
			mid=(l+h)/2;
			if(a[mid]==key)
				return mid;
			else if(a[mid]<key)
				l=mid+1;
			else
				h=mid-1;
		}

		return -1;
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		System.out.println("Enter array size:");
		int n = obj.nextInt();

		int i,a[]=new int[n];

		System.out.println("Enter "+n+" values:");
		for(i=0;i<n;i++)
			a[i] = obj.nextInt();

		System.out.println("Enter the value to search:");
		int key = obj.nextInt();

		Arrays.sort(a);

		System.out.println(Demo.binarysearch(a,key));
	}
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
```

```
Enter 6 values:
10 20 30 40 50 60
Enter the value to search:
10
0

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 values:
10 20 30 40 50 60
Enter the value to search:
60
5

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 values:
10 20 30 40 50 60
Enter the value to search:
90
-1
```

Ex:

```
import java.util.*;

class Demo
{
      static int binarysearch(int a[],int l,int h,int key){
            if(l<=h){
                  int mid=(l+h)/2;
                  if(a[mid]==key)
                        return mid;
                  else if(a[mid]<key)
                        return binarysearch(a,mid+1,h,key);
                  else
                        return binarysearch(a,l,mid-1,key);
            }
            return -1;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int n = obj.nextInt();

            int i,a[]=new int[n];
            System.out.println("Enter "+n+" values:");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();

            System.out.println("Enter the value to search:");
            int key = obj.nextInt();

            Arrays.sort(a);

            System.out.println(Demo.binarysearch(a,0,a.length-1,key));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
1 2 3 4 5
Enter the value to search:
2
1

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
1 2 3 4 5
Enter the value to search:
1
0
```

Sorting and Searching

Asc Order
Desc Order
Linear Search
Binary Search

Binary Search -----> 0 to n-1

Half Binary Search ----> First Half or Second Half or in between

Binary Search Version3

I want to search for element in the first half of the list i.e. 0 to n/2.

Ex:

```
import java.util.*;

class Demo
{
      static int binarysearch(int a[],int l,int h,int key){
            if(l<=h){
                  int mid=(l+h)/2;
                  if(a[mid]==key)
                        return mid;
                  else if(a[mid]<key)
                        return binarysearch(a,mid+1,h,key);
                  else
                        return binarysearch(a,l,mid-1,key);
            }
            return -1;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int n = obj.nextInt();

            int i,a[]=new int[n];

            System.out.println("Enter "+n+" values:");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();



            Arrays.sort(a);

            System.out.println("Array Elements after sorting...");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            System.out.println("Enter the value to search:");
            int key = obj.nextInt();

            System.out.println(Demo.binarysearch(a,0,(a.length-
1)/2,key));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java
```

```
C:\prakashclasses>java Test
Enter array size:
6
Enter 6 values:
11 12 13 16 14 15
Array Elements after sorting...
11 12 13 14 15 16
Enter the value to search:
11
0

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 values:
11 12 13 16 14 15
Array Elements after sorting...
11 12 13 14 15 16
Enter the value to search:
16
-1
```

Binary Search Version4

I want to search for element in the second half of the list i.e. n/2+1 to n.

Ex:

```
import java.util.*;

class Demo
{
      static int binarysearch(int a[],int l,int h,int key){
            if(l<=h){
                  int mid=(l+h)/2;
                  if(a[mid]==key)
                        return mid;
                  else if(a[mid]<key)
                        return binarysearch(a,mid+1,h,key);
                  else
                        return binarysearch(a,l,mid-1,key);
            }
            return -1;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int n = obj.nextInt();

            int i,a[]=new int[n];

            System.out.println("Enter "+n+" values:");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();



            Arrays.sort(a);

            System.out.println("Array Elements after sorting...");
            for(i=0;i<n;i++)
                  System.out.print(a[i]+" ");
            System.out.println();

            System.out.println("Enter the value to search:");
            int key = obj.nextInt();

            System.out.println(Demo.binarysearch(a,(a.length-
1)/2,a.length,key));
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 values:
11 14 12 15 13 16
Array Elements after sorting...
11 12 13 14 15 16
Enter the value to search:

11
-1

C:\prakashclasses>java Test
Enter array size:
6
Enter 6 values:
11 14 12 15 13 16
Array Elements after sorting...
11 12 13 14 15 16
Enter the value to search:
16
5
```

Binary Search Version5

I want to search for element in the between low and high values i.e. low to high.

```
low -----> inclusive
high ----> exclusive

Ex:
---
import java.util.*;

class Demo
{
    static int binarysearch(int a[],int l,int h,int key){
        if(l<=h){
            int mid=(l+h)/2;
            if(a[mid]==key)
                return mid;
            else if(a[mid]<key)
                return binarysearch(a,mid+1,h,key);
            else
                return binarysearch(a,l,mid-1,key);
        }
        return -1;
    }
}

class Test
{
    public static void main(String[] args)
    {
        Scanner obj = new Scanner(System.in);

        System.out.println("Enter array size:");
        int n = obj.nextInt();

        int i,a[]=new int[n];

        System.out.println("Enter "+n+" values:");
        for(i=0;i<n;i++)
            a[i] = obj.nextInt();



        Arrays.sort(a);

        System.out.println("Array Elements after sorting...");
        for(i=0;i<n;i++)
            System.out.println(i+"===>"+a[i]);

        System.out.println("Enter the value to search:");
        int key = obj.nextInt();

        System.out.println("Enter starting location:");
        int start = obj.nextInt();
```

```
            System.out.println("Enter ending location");
            int end = obj.nextInt();

            System.out.println(Demo.binarysearch(a,start,end,key));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
111 888 333 666 999
Array Elements after sorting...
0===>111
1===>333
2===>666
3===>888
4===>999
Enter the value to search:
666
Enter starting location:
1
Enter ending location
3
2

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
111 888 333 666 999
Array Elements after sorting...
0===>111
1===>333
2===>666
3===>888
4===>999
Enter the value to search:
666
Enter starting location:
3
Enter ending location
4
-1
```

Binary Search Version6

```
Arrays.sort(a)
Arrays.binarySearch(arrayname,start,end,key)
```

it will search for the given key inbetween start to end-1 in an array a.

```
Ex:
---
import java.util.*;


class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter array size:");
            int n = obj.nextInt();

            int i,a[]=new int[n];

            System.out.println("Enter "+n+" values:");
            for(i=0;i<n;i++)
                  a[i] = obj.nextInt();


            Arrays.sort(a);

            System.out.println("Array Elements after sorting...");
            for(i=0;i<n;i++)
                  System.out.println(i+"===>"+a[i]);

            System.out.println("Enter the value to search:");
            int key = obj.nextInt();

            System.out.println(Arrays.binarySearch(a,0,a.length,key));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
1 5 2 4 3
Array Elements after sorting...
0===>1
1===>2
2===>3
3===>4
4===>5
Enter the value to search:
```

```
5
4

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
1 5 2 4 3
Array Elements after sorting...
0===>1
1===>2
2===>3
3===>4
4===>5
Enter the value to search:
34
-6
```

Binary search version1: nomarl implementation

Binary search version2: recursive

Binary search version3: 0 to n/2

Binary search version4: n/2+1 to n

Binary search version5: start and ending location

Binary search version6: predefined method

Arrays.toString(array)

It will read elements one-by-one from an array and it converts into the following string format.

"["+e1+", "+e2+", "+e3+", "+...+"+en"]" ---> toString()

11,12,13,14 ----> [11, 12, 13, 14]

Ex:

```
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		System.out.println("Enter array size:");
		int n = obj.nextInt();

		int i,a[]=new int[n];

		System.out.println("Enter "+n+" values:");
		for(i=0;i<n;i++)
			a[i] = obj.nextInt();
```

```
            System.out.println("Array Elements one-by-one...");
            for(i=0;i<n;i++)
                  System.out.println(i+"===>"+a[i]);

            System.out.println(Arrays.toString(a));
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter array size:
5
Enter 5 values:
111
222
333
999
777
Array Elements one-by-one...
0===>111
1===>222
2===>333
3===>999
4===>777
[111, 222, 333, 999, 777]
```

equality of two arrays:
~~~~~~~~~~~~~~~~~~~~~~~

two arrays are said to equal iff each element present in the first array must be existed in the second also with same positions.

version1:

```
Ex1:
      [10, 20, 30]
      [11, 12, 13]
      false

Ex2:
      [10, 20, 30]
      [10, 11, 12]
      1st com--> ok
      2nd com
      false
Ex3:
      [10, 20, 30]
      [10, 20, 30]
      1sr com --> ok
      2nd com --> ok
```
~~~~~~~~~~~~~~~~~~~~~~~

      3rd com --> ok

```
{
      for(i=0;i<a.length;i++)
      {
            if(a[i]!=b[i])
                  return false;
      }
      return true;
}
```

Ex:
[10,20,30]
[20,10,30]
aren't these equal

w.r.t position ---> false
ignore position --> true (sort both arrays)

```
import java.util.*;

class Demo
{
      static boolean equals(int a[],int b[])
      {
            for(int i=0;i<a.length;i++)
            {
                  if(a[i]!=b[i])
                        return false;
            }
            return true;
      }
}

class Test
{
      public static void main(String[] args)
      {
            System.out.println(Demo.equals(new int[]{1,2,3},new
int[]{1,2,3}));//true
            System.out.println(Demo.equals(new int[]{1,2,3},new
int[]{4,5,6}));//false
            System.out.println(Demo.equals(new int[]{1,2,3},new
int[]{3,2,1}));//false
      }
}
```

output:

```
true
false
false
```

Ex:

```
import java.util.*;

class Demo
{
      static boolean equals(int a[],int b[])
      {
            for(int i=0;i<a.length;i++)
            {
                  if(a[i]!=b[i])
                        return false;
            }
            return true;
      }
}

class Test
{
      public static void main(String[] args)
      {
            System.out.println(Demo.equals(new int[]{1,2,3},new
int[]{1,2,3}));//true
            System.out.println(Demo.equals(new int[]{1,2,3},new
int[]{4,5,6}));//false
            System.out.println(Demo.equals(new int[]{1,2,3},new
int[]{3,2,1}));//false
            int a[] = {1,2,3};
            int b[] = {3,2,1};
            System.out.println(Demo.equals(a,b));//false
            Arrays.sort(a);
            Arrays.sort(b);
            System.out.println(Demo.equals(a,b));//true
      }
}
```

```
output:
-------
true
false
false
false
true
```

version2:
----

we have predefined method is existed for arrays comaprision

```
Arrays.equals(a,b);
```

Ex:

```
import java.util.*;

class Demo
{
}
```

```
class Test
{
      public static void main(String[] args)
      {
            int a[] = {1,2,3};
            int b[] = {3,2,1};
            System.out.println(Arrays.equals(a,b));//false
            Arrays.sort(a);
            Arrays.sort(b);
            System.out.println(Arrays.equals(a,b));//true
      }
}

output:
-------
false
true


inserting an element into an array
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
case1: inserting an element into an array at last position
----------------------------------------------------------
import java.util.*;

class Demo
{
      static int[] insertAtLast(int a[],int element)
      {
            int i,b[] = new int[a.length+1];
            for(i=0;i<a.length;i++)
            {
                  b[i] = a[i];
            }
            b[i] = element;
            return b;
      }
}

class Test
{
      public static void main(String[] args)
      {
            int a[] = {10,20,30,40,50};
            System.out.println(Arrays.toString(a));
            System.out.println(Arrays.toString(Demo.insertAtLast(a,60)));
      }
}

output:
-------
[10, 20, 30, 40, 50]
[10, 20, 30, 40, 50, 60]
```

case2: inserting an element at frist location

```
import java.util.*;

class Demo
{
	static int[] insertAtLast(int a[],int element)
	{
		int i,b[] = new int[a.length+1];
		for(i=0;i<a.length;i++)
		{
			b[i] = a[i];
		}
		b[i] = element;
		return b;
	}
	static int[] insertAtFirst(int a[],int element)
	{
		int i,b[] = new int[a.length+1];
		b[0] = element;
		for(i=0;i<a.length;i++)
		{
			b[i+1] = a[i];
		}
		return b;
	}
}

class Test
{
	public static void main(String[] args)
	{
		int a[] = {10,20,30,40,50};
		System.out.println(Arrays.toString(a));

	//System.out.println(Arrays.toString(Demo.insertAtLast(a,60)));
		System.out.println(Arrays.toString(Demo.insertAtFirst(a,5)));
	}
}
```

output:

```
[10, 20, 30, 40, 50]
[5, 10, 20, 30, 40, 50]
```

case3: inserting an element at given location

```
import java.util.*;

class Demo
{
	static int[] insertAtLast(int a[],int element)
```

```
    {
        int i,b[] = new int[a.length+1];
        for(i=0;i<a.length;i++)
        {
            b[i] = a[i];
        }
        b[i] = element;
        return b;
    }
    static int[] insertAtFirst(int a[],int element)
    {
        int i,b[] = new int[a.length+1];
        b[0] = element;
        for(i=0;i<a.length;i++)
        {
            b[i+1] = a[i];
        }
        return b;
    }
    static int[] insertAtLocation(int a[],int element,int location)
    {
        int i,k=0,b[] = new int[a.length+1];
        for(i=0;i<location;i++)
            b[k++]=a[i];
        b[k++]=element;
        for(i=location;i<a.length;i++)
            b[k++]=a[i];
        return b;
    }
}

class Test
{
    public static void main(String[] args)
    {
        int a[] = {10,20,30,40,50};
        System.out.println(Arrays.toString(a));

    //System.out.println(Arrays.toString(Demo.insertAtLast(a,60)));

    //System.out.println(Arrays.toString(Demo.insertAtFirst(a,5)));

    System.out.println(Arrays.toString(Demo.insertAtLocation(a,999,0)))
;

    System.out.println(Arrays.toString(Demo.insertAtLocation(a,999,1)))
;

    System.out.println(Arrays.toString(Demo.insertAtLocation(a,999,2)))
;

    System.out.println(Arrays.toString(Demo.insertAtLocation(a,999,3)))
;
```

```
      System.out.println(Arrays.toString(Demo.insertAtLocation(a,999,4)))
;
      }
}
```

```
output:
-------
[10, 20, 30, 40, 50]
[999, 10, 20, 30, 40, 50]
[10, 999, 20, 30, 40, 50]
[10, 20, 999, 30, 40, 50]
[10, 20, 30, 999, 40, 50]
[10, 20, 30, 40, 999, 50]
```

```
DELETING AN ELEMENT FROM AN ARRAY
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
case1: deleting an element located at the given location
--------------------------------------------------------
import java.util.*;

class Demo
{
      static int[] deleteElementAtLocation(int a[],int location)
      {
            int k=0,i,b[] = new int[a.length-1];
            for(i=0;i<a.length;i++)
            {
                  if(i==location)
                        continue;
                  b[k++]=a[i];
            }
            return b;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {10,11,12,13,14,15};
            System.out.println(Arrays.toString(a));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,0
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,1
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,2
)));
```

```
      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,3
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,4
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,5
)));
      }
}


output:
-------
[10, 11, 12, 13, 14, 15]
[11, 12, 13, 14, 15]
[10, 12, 13, 14, 15]
[10, 11, 13, 14, 15]
[10, 11, 12, 14, 15]
[10, 11, 12, 13, 15]
[10, 11, 12, 13, 14]

case2: delete all elements in an array
---------------------------------------
import java.util.*;

class Demo
{
      static int[] deleteElementAtLocation(int a[],int location)
      {
            int k=0,i,b[] = new int[a.length-1];
            for(i=0;i<a.length;i++)
            {
                  if(i==location)
                        continue;
                  b[k++]=a[i];
            }
            return b;
      }
      static int[] deleteAll(int a[]){
            int b[]=new int[0];
            return b;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {10,11,12,13,14,15};
            System.out.println(Arrays.toString(a));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,0
)));
```

```
      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,1
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,2
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,3
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,4
)));

      System.out.println(Arrays.toString(Demo.deleteElementAtLocation(a,5
)));
            System.out.println(Arrays.toString(Demo.deleteAll(a)));
      }
}
```

```
output:
-------
[10, 11, 12, 13, 14, 15]
[11, 12, 13, 14, 15]
[10, 12, 13, 14, 15]
[10, 11, 13, 14, 15]
[10, 11, 12, 14, 15]
[10, 11, 12, 13, 15]
[10, 11, 12, 13, 14]
[]
```

```
case3: deleting an element from an array
----------------------------------------
import java.util.*;

class Demo
{
      static int[] deleteElementAtLocation(int a[],int location)
      {
            int k=0,i,b[] = new int[a.length-1];
            for(i=0;i<a.length;i++)
            {
                  if(i==location)
                        continue;
                  b[k++]=a[i];
            }
            return b;
      }
      static int[] deleteAll(int a[]){
            a=new int[0];
            return a;
      }
      static int[] deleteElement(int a[],int element)
      {
            int index=-1,i,k=0;
            for(i=0;i<a.length;i++)
            {
                  if(a[i]==element)
                  {
                        index=i;
                        break;
                  }
            }
            if(index!=-1)
            {
                  int b[] = new int[a.length-1];
                  for(i=0;i<a.length;i++)
                  {
                        if(i==index)
                              continue;
                        b[k++]=a[i];
                  }
                  return b;
            }
            return a;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {10,11,12,13,14,15};
            System.out.println(Arrays.toString(a));

      System.out.println(Arrays.toString(Demo.deleteElement(a,10)));

      System.out.println(Arrays.toString(Demo.deleteElement(a,11)));

      System.out.println(Arrays.toString(Demo.deleteElement(a,12)));

      System.out.println(Arrays.toString(Demo.deleteElement(a,13)));

      System.out.println(Arrays.toString(Demo.deleteElement(a,14)));

      System.out.println(Arrays.toString(Demo.deleteElement(a,15)));

      }
}
```

```
output:
-------
[10, 11, 12, 13, 14, 15]
[11, 12, 13, 14, 15]
[10, 12, 13, 14, 15]
[10, 11, 13, 14, 15]
[10, 11, 12, 14, 15]
[10, 11, 12, 13, 15]
```

```
[10, 11, 12, 13, 14]
```

```
UPDATING ELEMENT IN AN ARRAY
----------------------------
case1: updating based on location
----------------------------------
import java.util.*;

class Demo
{
      static int[] updateElementAtLocation(int a[],int location,int
element)
      {
            int b[] = new int[a.length];
            for(int i=0;i<a.length;i++)
                  b[i] = a[i];
            if(location>=0 && location<a.length)
                  b[location]=element;
            return b;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {10,11,12,13,14,15};
            System.out.println(Arrays.toString(a));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,0
,999)));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,1
,999)));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,2
,999)));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,3
,999)));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,4
,999)));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,5
,999)));

      System.out.println(Arrays.toString(Demo.updateElementAtLocation(a,7
,999)));
      }
}
```

output:

```
-------
[10, 11, 12, 13, 14, 15]
[999, 11, 12, 13, 14, 15]
[10, 999, 12, 13, 14, 15]
[10, 11, 999, 13, 14, 15]
[10, 11, 12, 999, 14, 15]
[10, 11, 12, 13, 999, 15]
[10, 11, 12, 13, 14, 999]
[10, 11, 12, 13, 14, 15]
```

case2: update based on elemenet:

```
--------------------------------
import java.util.*;

class Demo
{
      static int[] updateElementAtLocation(int a[],int location,int 
element)
      {
            int b[] = new int[a.length];
            for(int i=0;i<a.length;i++)
                  b[i] = a[i];
            if(location>=0 && location<a.length)
                  b[location]=element;
            return b;
      }
      static int[] updateElement(int a[],int oldElement,int newElement)
      {
            int i,b[] = new int[a.length];
            for(i=0;i<a.length;i++)
                  b[i] = a[i];
            for(i=0;i<b.length;i++)
            {
                  if(b[i]==oldElement)
                  {
                        b[i] = newElement;
                        break;
                  }
            }
            return b;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {10,11,12,13,14,15};
            System.out.println(Arrays.toString(a));

      System.out.println(Arrays.toString(Demo.updateElement(a,10,999)));

      System.out.println(Arrays.toString(Demo.updateElement(a,90,999)));
      }
}
```

```
output:
-------
[10, 11, 12, 13, 14, 15]
[999, 11, 12, 13, 14, 15]
[10, 11, 12, 13, 14, 15]
```

two-d arrays:

==> row and cols
==> two-d arrays not implemented in 'array of arrays' style

Implement a program to read and write matrix element

```
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		System.out.println("Enter matrix row size:");
		int rsize = obj.nextInt();

		System.out.println("Enter matrix column size:");
		int csize = obj.nextInt();

		int i,j,a[][] = new int[rsize][csize];

		System.out.println("Enter matrix element one-by-one:");
		for(i=0;i<rsize;i++)
		{
			for(j=0;j<csize;j++)
			{
				a[i][j] = obj.nextInt();
			}
		}

		System.out.println("MATRIX ELEMENTS ARE:");
		for(i=0;i<rsize;i++)
		{
			for(j=0;j<csize;j++)
			{
				System.out.print(a[i][j]+"["+i+","+j+"] ");
			}
			System.out.println();
		}
	}
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 2 3
4 5 6
7 8 9
MATRIX ELEMENTS ARE:
1[0,0] 2[0,1] 3[0,2]
4[1,0] 5[1,1] 6[1,2]
7[2,0] 8[2,1] 9[2,2]
```

Implement a program to perform addition of two matrices

```
--------------------------------------------------------
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter matrix-A row size:");
            int rsize1 = obj.nextInt();

            System.out.println("Enter matrix-A column size:");
            int csize1 = obj.nextInt();


            System.out.println("Enter matrix-B row size:");
            int rsize2 = obj.nextInt();

            System.out.println("Enter matrix-B column size:");
            int csize2 = obj.nextInt();

            if(rsize1==rsize2 && csize1==csize2)
            {
                  int i,j;
                  int a[][] = new int[rsize1][csize1];
                  int b[][] = new int[rsize2][csize2];
                  int c[][] = new int[rsize1][csize1];

                  System.out.println("Enter matrix-A element one-by-
one:");
                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize1;j++)
                        {
                              a[i][j] = obj.nextInt();
                        }
                  }

                  System.out.println("Enter matrix-B element one-by-one:");
                  for(i=0;i<rsize2;i++)
                  {
                        for(j=0;j<csize2;j++)
                        {
                              b[i][j] = obj.nextInt();
                        }
                  }

                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize1;j++)
                        {
                              c[i][j] = a[i][j] + b[i][j];
                        }
                  }

                  System.out.println("MATRIX-A ELEMENTS ARE:");
                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize1;j++)
                        {
                              System.out.print(a[i][j]+" ");
                        }
                        System.out.println();
                  }

                  System.out.println("MATRIX-B ELEMENTS ARE:");
                  for(i=0;i<rsize2;i++)
                  {
                        for(j=0;j<csize2;j++)
                        {
                              System.out.print(b[i][j]+" ");
                        }
                        System.out.println();
                  }

                  System.out.println("MATRIX-C ELEMENTS ARE:");
                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize1;j++)
                        {
                              System.out.print(c[i][j]+" ");
                        }
                        System.out.println();
                  }
```

```
            }
            else
            {
                  System.out.println("MATRIX addition is not possible");
            }
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix-A row size:
3
Enter matrix-A column size:
3
Enter matrix-B row size:
4
Enter matrix-B column size:
5
MATRIX addition is not possible

C:\prakashclasses>java Test
Enter matrix-A row size:
3
Enter matrix-A column size:
3
Enter matrix-B row size:
3
Enter matrix-B column size:
3
Enter matrix-A element one-by-one:
1 2 3
4 5 6
7 8 9
Enter matrix-B element one-by-one:
1 2 3
4 5 6
7 8 9
MATRIX-A ELEMENTS ARE:
1 2 3
4 5 6
7 8 9
MATRIX-B ELEMENTS ARE:
1 2 3
4 5 6
7 8 9
MATRIX-C ELEMENTS ARE:
2 4 6
8 10 12
14 16 18
```

```
multiplication:
---------------
scalar matrix multiplication
two matrix multiplication

Ex:
1 2 3
4 5 6
7 8 9

2

2 4 6
8 10 12
14 16 18

Ex:
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter matrix-A row size:");
            int rsize1 = obj.nextInt();

            System.out.println("Enter matrix-A column size:");
            int csize1 = obj.nextInt();


            System.out.println("Enter matrix-B row size:");
            int rsize2 = obj.nextInt();

            System.out.println("Enter matrix-B column size:");
            int csize2 = obj.nextInt();

            if(rsize1==csize2)
            {
                  int i,j,k;
                  int a[][] = new int[rsize1][csize1];
                  int b[][] = new int[rsize2][csize2];
                  int c[][] = new int[rsize1][csize1];

                  System.out.println("Enter matrix-A element one-by-one:");
                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize1;j++)
                        {
                              a[i][j] = obj.nextInt();
                        }
                  }
                  System.out.println("Enter matrix-B element one-by-
one:");
                  for(i=0;i<rsize2;i++)
                  {
                        for(j=0;j<csize2;j++)
                        {
                              b[i][j] = obj.nextInt();
                        }
                  }

                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize2;j++)
                        {
                              c[i][j] = 0;
                              for(k=0;k<csize1;k++)
                              {
                                    c[i][j] = c[i][j] + (a[i][k]*b[k][j]);
                              }
                        }
                  }

                  System.out.println("MATRIX-A ELEMENTS ARE:");
                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize1;j++)
                        {
                              System.out.print(a[i][j]+" ");
                        }
                        System.out.println();
                  }

                  System.out.println("MATRIX-B ELEMENTS ARE:");
                  for(i=0;i<rsize2;i++)
                  {
                        for(j=0;j<csize2;j++)
                        {
                              System.out.print(b[i][j]+" ");
                        }
                        System.out.println();
                  }

                  System.out.println("MATRIX-C ELEMENTS ARE:");
                  for(i=0;i<rsize1;i++)
                  {
                        for(j=0;j<csize2;j++)
                        {
                              System.out.print(c[i][j]+" ");
                        }
                        System.out.println();
                  }
            }
            else
            {
                  System.out.println("MATRIX MULTIPLICATION is not 
possible");
            }
      }
}
```

```
output:
-------
C:\prakashclasses>java Test
Enter matrix-A row size:
3
Enter matrix-A column size:
3
Enter matrix-B row size:
3
Enter matrix-B column size:
3
Enter matrix-A element one-by-one:
1 2 3
4 5 6
7 8 9
Enter matrix-B element one-by-one:
1 0 0
0 1 0
0 0 1
MATRIX-A ELEMENTS ARE:
1 2 3
4 5 6
7 8 9
MATRIX-B ELEMENTS ARE:
1 0 0
0 1 0
0 0 1
MATRIX-C ELEMENTS ARE:
1 2 3
4 5 6
7 8 9
```

01) Program to read and write matrix elements.
02) Program to read and calcualte addition of two matrices.
03) Program to read and calcualte subtraction of two matrices.
04) Program to perform scalar matrix multiplication.
05) Program to perform normal matrix multiplication.

06) Program to read and calcualte sum of all the elements present in the matrix
----

```
import java.util.*;
```

```
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter matrix row size:");
            int rsize = obj.nextInt();

            System.out.println("Enter matrix column size:");
            int csize = obj.nextInt();


            int i,j,sum;
            int a[][] = new int[rsize][csize];

            System.out.println("Enter matrix element one-by-one:");
            for(i=0;i<rsize;i++)
            {
                  for(j=0;j<csize;j++)
                  {
                        a[i][j] = obj.nextInt();
                  }
            }

            sum=0;
            for(i=0;i<rsize;i++)
            {
                  for(j=0;j<csize;j++)
                  {
                        sum=sum+a[i][j];
                  }
            }
            System.out.println("Sum ="+sum);

      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 2 3
4 5 6
7 8 9
Sum =45
```

07) Program to find row wise sum values

```
---------------------------------------
Ex:
---
1 2 3 ----> 6
4 5 6 ----> 15
7 8 9 ----> 24

output:
-------
6
15
24

Ex:
----
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter matrix row size:");
            int rsize = obj.nextInt();

            System.out.println("Enter matrix column size:");
            int csize = obj.nextInt();


            int i,j,sum;
            int a[][] = new int[rsize][csize];

            System.out.println("Enter matrix element one-by-one:");
            for(i=0;i<rsize;i++)
            {
                  for(j=0;j<csize;j++)
                  {
                        a[i][j] = obj.nextInt();
                  }
            }

            for(i=0;i<rsize;i++)
            {
                  sum=0;
                  for(j=0;j<csize;j++)
                  {
                        sum=sum+a[i][j];
                  }
                  System.out.println((i+1)+"Row Sum= "+sum);
            }
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 2 3
4 5 6
7 8 9
1Row Sum= 6
2Row Sum= 15
3Row Sum= 24
```

08) Program to find col wise sum values

Ex:

```
1 2 3 ----> 6
4 5 6 ----> 15
7 8 9 ----> 24
```

Ex:

```
1 4 7 ----> 12
2 5 8 ----> 15
3 6 9 ----> 18
```

```
a[i][j] ----> a[j][i]
```

Ex:

```
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		System.out.println("Enter matrix row size:");
		int rsize = obj.nextInt();

		System.out.println("Enter matrix column size:");
		int csize = obj.nextInt();


		int i,j,sum;
		int a[][] = new int[rsize][csize];
```

```
        System.out.println("Enter matrix element one-by-one:");
        for(i=0;i<rsize;i++)
        {
            for(j=0;j<csize;j++)
            {
                a[i][j] = obj.nextInt();
            }
        }

        for(i=0;i<rsize;i++)
        {
            sum=0;
            for(j=0;j<csize;j++)
            {
                sum=sum+a[j][i];
            }
            System.out.println((i+1)+" Col Sum= "+sum);
        }
    }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 2 3
4 5 6
7 8 9
1 Col Sum= 12
2 Col Sum= 15
3 Col Sum= 18
```

```
09) Program to calcualte tranpose of the given matrix
-----------------------------------------------------
Ex:
---
1 2 3
4 5 6
7 8 9

Original Matrix:
----------------
1 2 3
4 5 6
7 8 9

Transpose Matrix:
-----------------
```

```
1 4 7
2 5 8
3 6 9

a[i][j] ---> a[j][i]

Ex:
---
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		System.out.println("Enter matrix row size:");
		int rsize = obj.nextInt();

		System.out.println("Enter matrix column size:");
		int csize = obj.nextInt();


		int i,j;
		int a[][] = new int[rsize][csize];
		int b[][] = new int[rsize][csize];

		System.out.println("Enter matrix element one-by-one:");
		for(i=0;i<rsize;i++)
		{
			for(j=0;j<csize;j++)
			{
				a[i][j] = obj.nextInt();
			}
		}
		for(i=0;i<rsize;i++)
		{
			for(j=0;j<csize;j++)
			{
				b[i][j]=a[j][i];
			}
		}

		System.out.println("Original Matrix Elements:");
		for(i=0;i<rsize;i++)
		{
			for(j=0;j<csize;j++)
			{
				System.out.print(a[i][j]+" ");
			}
			System.out.println();
		}
		System.out.println("Trnaspose Matrix Elements:");
		for(i=0;i<rsize;i++)
```

```
            {
                  for(j=0;j<csize;j++)
                  {
                        System.out.print(b[i][j]+" ");
                  }
                  System.out.println();
            }
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 2 3
4 5 6
7 8 9
Original Matrix Elements:
1 2 3
4 5 6
7 8 9
Trnaspose Matrix Elements:
1 4 7
2 5 8
3 6 9
```

10) Program to check whether the given matrix is identity matrix or not?

Format:

```
            diagonal elements should be '1'
            non-diagonal elements should be '0'

            1 0 0
            0 1 0
            0 0 1
```

Ex:

```
import java.util.*;

class Demo
{
      static boolean isIdentity(int a[][],int n,int m)
      {
            int i,j;
            for(i=0;i<n;i++)
            {
                  for(j=0;j<m;j++)
                  {
                        if(i!=j && a[i][j]!=0)
                              return false;
                        if(i==j && a[i][j]!=1)
                              return false;
                  }
            }
            return true;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter matrix row size:");
            int rsize = obj.nextInt();

            System.out.println("Enter matrix column size:");
            int csize = obj.nextInt();


            int i,j;
            int a[][] = new int[rsize][csize];

            System.out.println("Enter matrix element one-by-one:");
            for(i=0;i<rsize;i++)
            {
                  for(j=0;j<csize;j++)
                  {
                        a[i][j] = obj.nextInt();
                  }
            }

            System.out.println(Demo.isIdentity(a,rsize,csize));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 0 0
0 1 0
0 0 1
```

```
true

C:\prakashclasses>java Test
Enter matrix row size:
3
Enter matrix column size:
3
Enter matrix element one-by-one:
1 2 3
4 1 6
7 8 1
false
```

11) Swaping of two rows

Ex:

```
     1 2 3
     4 5 6
     7 8 9

     1st and 3rd row

     7 8 9
     4 5 6
     1 2 3
```

Logic:

```
          t = a[m-1][i]
          a[m-1][i] = a[n-1][i];
          a[n-1][i] = t;

          where m and n are rows to be interchanged..
```

Ex:

```
import java.util.*;

class Test
{
     public static void main(String[] args)
     {
          Scanner obj = new Scanner(System.in);

          System.out.println("Enter row value:");
          int row = obj.nextInt();

          System.out.println("Enter col value:");
          int col = obj.nextInt();
```

```
            int a[][] = new int[row][col];

            int i,j,n,m,t;

            System.out.println("Enter matrix elements:");

            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        a[i][j] = obj.nextInt();
                  }
            }

            System.out.println("Enter m and n values:");
            m=obj.nextInt();
            n=obj.nextInt();

            System.out.println("Before swaping:");
            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        System.out.print(a[i][j]+" ");
                  }
                  System.out.println();
            }

            for(i=0;i<col;i++){
                  t=a[m-1][i];
                  a[m-1][i]=a[n-1][i];
                  a[n-1][i]=t;
            }

            System.out.println("After swaping:");
            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        System.out.print(a[i][j]+" ");
                  }
                  System.out.println();
            }
      }
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter row value:
3
Enter col value:
3
Enter matrix elements:
1 2 3
4 5 6
7 8 9
Enter m and n values:
```

```
1
3
Before swaping:
1 2 3
4 5 6
7 8 9
After swaping:
7 8 9
4 5 6
1 2 3


11) Swaping of two cols
~~~~~~~~~~~~~~~~~~~~~~~
Ex:
---
      1 2 3
      4 5 6
      7 8 9

      1st and 3rd col

      3 2 1
      6 5 4
      9 8 7

Logic:
------
            t = a[i][m-1]
            a[i][m-1] = a[i][n-1];
            a[i][n-1] = t;

            where m and n are rows to be interchanged..

Ex:
---
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter row value:");
            int row = obj.nextInt();

            System.out.println("Enter col value:");
            int col = obj.nextInt();

            int a[][] = new int[row][col];

            int i,j,n,m,t;
```

```
            System.out.println("Enter matrix elements:");

            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        a[i][j] = obj.nextInt();
                  }
            }

            System.out.println("Enter m and n values:");
            m=obj.nextInt();
            n=obj.nextInt();

            System.out.println("Before swaping:");
            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        System.out.print(a[i][j]+" ");
                  }
                  System.out.println();
            }

            for(i=0;i<col;i++){
                  t=a[i][m-1];
                  a[i][m-1]=a[i][n-1];
                  a[i][n-1]=t;
            }

            System.out.println("After swaping:");
            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        System.out.print(a[i][j]+" ");
                  }
                  System.out.println();
            }
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter row value:
4
Enter col value:
4
Enter matrix elements:
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
Enter m and n values:
1
3
Before swaping:
```

```
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
After swaping:
0 0 1 0
0 1 0 0
1 0 0 0
0 0 0 1
```

13) sum of diagonal elements

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ex:
---
            1 2 3
            4 5 6
            7 8 9

            1,5,9 ---> 1+5+9=15

Logic:
------
            s=0;
            for(i=0;i<row;i++)
            {
                  for(j=0;j<col;j++)
                  {
                        if(i==j)
                        {
                              s=s+a[i][j];
                        }
                  }
            }
            print s

Ex:
---
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter row value:");
            int row = obj.nextInt();

            System.out.println("Enter col value:");
            int col = obj.nextInt();

            int a[][] = new int[row][col];
```

```
            int i,j,s;

            System.out.println("Enter matrix elements:");

            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        a[i][j] = obj.nextInt();
                  }
            }
            s=0;
            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        if(i==j)
                        {
                              s=s+a[i][j];
                        }
                  }
            }
            System.out.println(s);
      }
}

output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter row value:
3
Enter col value:
3
Enter matrix elements:
1 2 3
4 5 6
7 8 9
15

14) sum of opposite diagonal elements
-------------------------------------
Ex:
---
            1 2 3
            4 5 6
            7 8 9

            3,5,7 ---> 3+5+7=15

Logic:
------
            main dia ----> a[i][i]
            opp dia -----> a[i][n-i-1]

Ex:
---
```

```
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		System.out.println("Enter row value:");
		int row = obj.nextInt();

		System.out.println("Enter col value:");
		int col = obj.nextInt();

		int a[][] = new int[row][col];

		int i,j,s;

		System.out.println("Enter matrix elements:");

		for(i=0;i<row;i++){
			for(j=0;j<col;j++){
				a[i][j] = obj.nextInt();
			}
		}
		s=0;
		for(i=0;i<row;i++){
			s=s+a[i][row-i-1];
		}
		System.out.println(s);
	}
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter row value:
3
Enter col value:
3
Enter matrix elements:
1 2 3
4 5 6
7 8 9
15

C:\prakashclasses>java Test
Enter row value:
4
Enter col value:
4
Enter matrix elements:
```

```
1 2 3 4
5 6 7 8
9 1 2 3
1 0 0 1
13
```

15) interchanging of diagonal

Ex:

```
      1 2 3
      4 5 6
      7 8 9

      main dia and opp dia

      3 2 1
      4 5 6
      9 8 7

      a[i][i]
      a[i][n-i-1]

      t=a[i][i]
      a[i][i]=a[i][n-i-1]
      a[i][n-i-1]=t;
```

Ex:
--

```
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            System.out.println("Enter row value:");
            int row = obj.nextInt();

            System.out.println("Enter col value:");
            int col = obj.nextInt();

            int a[][] = new int[row][col];

            int i,j,t;

            System.out.println("Enter matrix elements:");

            for(i=0;i<row;i++){
                  for(j=0;j<col;j++){
                        a[i][j] = obj.nextInt();
                  }
```

```
			}

			System.out.println("Before swaping...");
			for(i=0;i<row;i++){
				for(j=0;j<col;j++){
					System.out.print(a[i][j]+" ");
				}
				System.out.println();
			}
			for(i=0;i<row;i++){
				t=a[i][i];
				a[i][i]=a[i][row-i-1];
				a[i][row-i-1]=t;
			}
			System.out.println("After swaping...");
			for(i=0;i<row;i++){
				for(j=0;j<col;j++){
					System.out.print(a[i][j]+" ");
				}
				System.out.println();
			}
		}
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter row value:
3
Enter col value:
3
Enter matrix elements:
1 2 3
4 5 6
7 8 9
Before swaping...
1 2 3
4 5 6
7 8 9
After swaping...
3 2 1
4 5 6
9 8 7
```

Array Rotations:
~~~~~~~~~~~~~~~~
Ex:

```
	[1, 2, 3, 4, 5] ---> Left Rotations
```
~~~~~~~~~~~~~~~~

```
                                   [2, 3, 4, 5, 1] ---> 1
                                   [3, 4, 5, 1, 2] ---> 2
                                   [4, 5, 1, 2, 3] ---> 3
                                   [5, 1, 2, 3, 4] ---> 4
                                   [1, 2, 3, 4, 5] ---> 5

      [1, 2, 3, 4, 5] ---> Right Rotations

                                   [5, 1, 2, 3, 4] ---> 1
                                   [4, 5, 1, 2, 3] ---> 2
                                   [3, 4, 5, 1, 2] ---> 3
                                   [2, 3, 4, 5, 1] ---> 4
```

we can perform these rotations in the following ways

1) Brute Force

Rotate all the elements by one position towards left/right direction for 'r' rotations.

```
n----> array size
a----> array
r----> number of rotations
```

Note:

```
            r=r%n
            n=5, r=1 -----> r=0 -------------------> r=1%5=1
            n=5, r=2 -----> r=0,1 -----------------> r=2%5=2
            n=5, r=3 -----> r=0,1,2 ---------------> r=3%5=3
            n=5, r=4 -----> r=0,1,2,3 -------------> r=4%5=4
            n=5, r=5 -----> loop wn't execute -----> r=5%5=0
            n=5, r=6 -----> r=0            --------> r=6%5=1
```

Ex:

```
import java.util.*;

class Demo
{
      static int[] rotateLeft(int a[],int r)
      {
            int temp,prev,i,j;
            for(i=0;i<r;i++)
            {
                  prev=a[0];
                  for(j=a.length-1;j>=0;j--){
                        temp=a[j];
                        a[j]=prev;
                        prev=temp;
                  }
            }
            return a;
      }
}


class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            int a[] = {1, 2, 3, 4, 5};

            System.out.println("Enter number of rotations(r):");
            int r = obj.nextInt();

            System.out.println("Before Rotation==>"+Arrays.toString(a));
            a=Demo.rotateLeft(a,r);
            System.out.println("After Rotation ==>"+Arrays.toString(a));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

```
C:\prakashclasses>java Test
Enter number of rotations(r):
6
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]
```

```
Ex;
---
import java.util.*;

class Demo
{
     static int[] rotateLeft(int a[],int r)
     {
          int temp,prev,i,j;
          for(i=0;i<r;i++)
          {
               prev=a[0];
               for(j=a.length-1;j>=0;j--){
                    temp=a[j];
                    a[j]=prev;
                    prev=temp;
               }
          }
          return a;
     }
     static int[] rotateRight(int a[],int r)
     {
          int temp,prev,i,j;
          for(i=0;i<r;i++)
          {
               prev=a[a.length-1];
               for(j=0;j<a.length;j++){
                    temp=a[j];
                    a[j]=prev;
                    prev=temp;
               }
          }
          return a;
     }
}


class Test
{
     public static void main(String[] args)
     {
          Scanner obj = new Scanner(System.in);

          int a[] = {1, 2, 3, 4, 5};

          System.out.println("Enter number of rotations(r):");
          int r = obj.nextInt();
```

```
            System.out.println("Before Rotation==>"+Arrays.toString(a));
            a=Demo.rotateRight(a,r);
            System.out.println("After Rotation ==>"+Arrays.toString(a));
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

2) by using 'temp' variable

Ex:

```
import java.util.*;

class Demo
{
      static int[] rotateLeft_Temp(int a[],int r)
      {
            r=r%a.length;
            int temp, i, j;

            for(i=0;i<r;i++)
            {
                  temp=a[0];
                  for(j=0;j<a.length-1;j++){
                        a[j]=a[j+1];
                  }
                  a[a.length-1]=temp;
            }

            return a;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            int a[] = {1, 2, 3, 4, 5};

            System.out.println("Enter number of rotations(r):");
            int r = obj.nextInt();

            System.out.println("Before Rotation==>"+Arrays.toString(a));
            a=Demo.rotateLeft_Temp(a,r);
            System.out.println("After Rotation ==>"+Arrays.toString(a));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]
```

```
C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

Ex:

```
import java.util.*;

class Demo
{
	static int[] rotateRight_Temp(int a[],int r)
	{
		r=r%a.length;

		int temp, i, j;

		for(i=0;i<r;i++)
		{
			temp=a[a.length-1];
			for(j=a.length-1;j>0;j--){
				a[j]=a[j-1];
			}
			a[0]=temp;
		}

		return a;
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		int a[] = {1, 2, 3, 4, 5};

		System.out.println("Enter number of rotations(r):");
		int r = obj.nextInt();

		System.out.println("Before Rotation==>"+Arrays.toString(a));
		a=Demo.rotateRight_Temp(a,r);
		System.out.println("After Rotation ==>"+Arrays.toString(a));
	}
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

3) by using temp array method-1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Ex:

```
import java.util.*;

class Demo
{
      static int[] rotateLeft_TempM1(int a[],int r)
      {
            r=r%a.length;
            int i,j,n=a.length;
            int temp[] = new int[r];

            for(i=0;i<r;i++)
                  temp[i]=a[i];

            for(i=r;i<n;i++)
                  a[i-r]=a[i];
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
            for(i=0;i<r;i++)
                  a[i+n-r]=temp[i];

            return a;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            int a[] = {1, 2, 3, 4, 5};

            System.out.println("Enter number of rotations(r):");
            int r = obj.nextInt();

            System.out.println("Before Rotation==>"+Arrays.toString(a));
            a=Demo.rotateLeft_TempM1(a,r);
            System.out.println("After Rotation ==>"+Arrays.toString(a));
      }
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
```

```
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

```
Ex:
---
import java.util.*;

class Demo
{
      static int[] rotateRight_TempM1(int a[],int r)
      {
            r=r%a.length;
            int i,j,n=a.length;
            int temp[] = new int[r];

            for(i=0;i<r;i++)
                  temp[i]=a[n-r+i];

            for(i=n-r-1;i>=0;i--)
                  a[i+r]=a[i];

            for(i=0;i<r;i++)
                  a[i]=temp[i];

            return a;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);

            int a[] = {1, 2, 3, 4, 5};

            System.out.println("Enter number of rotations(r):");
            int r = obj.nextInt();

            System.out.println("Before Rotation==>"+Arrays.toString(a));
            a=Demo.rotateRight_TempM1(a,r);
            System.out.println("After Rotation ==>"+Arrays.toString(a));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
```

```
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]

C:\prakashclasses>java Test
Enter number of rotations(r):
6
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]
```

```
4) by using temp array method-2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ex:
---
import java.util.*;

class Demo
{
      static int[] rotateLeft_TempM2(int a[],int r){
            r=r%a.length;
            int i,n=a.length;

            int temp[] = new int[n];

            for(i=0;i<n;i++)
                  temp[i] = a[(i+r)%n];

            for(i=0;i<n;i++)
                  a[i] = temp[i];

            return a;
```

```
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		int a[] = {1, 2, 3, 4, 5};

		System.out.println("Enter number of rotations(r):");
		int r = obj.nextInt();

		System.out.println("Before Rotation==>"+Arrays.toString(a));
		a=Demo.rotateLeft_TempM2(a,r);
		System.out.println("After Rotation ==>"+Arrays.toString(a));
	}
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

Ex:

```
import java.util.*;

class Demo
{
	static int[] rotateRight_TempM2(int a[],int r){
		r=r%a.length;
		int i,n=a.length;

		int temp[] = new int[n];

		for(i=0;i<n;i++)
			temp[(i+r)%n] = a[i];

		for(i=0;i<n;i++)
			a[i] = temp[i];

		return a;
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		int a[] = {1, 2, 3, 4, 5};

		System.out.println("Enter number of rotations(r):");
		int r = obj.nextInt();

		System.out.println("Before Rotation==>"+Arrays.toString(a));
		a=Demo.rotateRight_TempM2(a,r);
		System.out.println("After Rotation ==>"+Arrays.toString(a));
	}
}
```

output:

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]
```

```
C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]

C:\prakashclasses>java Test
Enter number of rotations(r):
6
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]
```

```
5) by using reversal algorithm
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ex:
---
import java.util.*;

class Demo
{
      static void reverse(int a[],int s,int e){
            int temp;
            while(s<e){
                  temp=a[s];
                  a[s]=a[e];
                  a[e]=temp;
                  s++;
                  e--;
            }
      }
      static int[] rotateLeft_reversal(int a[],int r){
            r=r%a.length;
            reverse(a,0,r-1);
            reverse(a,r,a.length-1);
            reverse(a,0,a.length-1);
            return a;
      }
}

class Test
```

```
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		int a[] = {1, 2, 3, 4, 5};

		System.out.println("Enter number of rotations(r):");
		int r = obj.nextInt();

		System.out.println("Before Rotation==>"+Arrays.toString(a));
		a=Demo.rotateLeft_reversal(a,r);
		System.out.println("After Rotation ==>"+Arrays.toString(a));
	}
}

output:
-------
C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]

Ex:
---
import java.util.*;

class Demo
{
	static void reverse(int a[],int s,int e){
		int temp;
		while(s<e){
			temp=a[s];
			a[s]=a[e];
			a[e]=temp;
			s++;
			e--;
		}
	}
	static int[] rotateRight_reversal(int a[],int r){
		r=r%a.length;
		reverse(a,0,a.length-1);
		reverse(a,0,r-1);
		reverse(a,r,a.length-1);
		return a;
	}
}

class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);

		int a[] = {1, 2, 3, 4, 5};

		System.out.println("Enter number of rotations(r):");
		int r = obj.nextInt();

		System.out.println("Before Rotation==>"+Arrays.toString(a));
		a=Demo.rotateRight_reversal(a,r);
		System.out.println("After Rotation ==>"+Arrays.toString(a));
	}
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Enter number of rotations(r):
1
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[5, 1, 2, 3, 4]

C:\prakashclasses>java Test
Enter number of rotations(r):
2
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[4, 5, 1, 2, 3]

C:\prakashclasses>java Test
Enter number of rotations(r):
```

```
3
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[3, 4, 5, 1, 2]

C:\prakashclasses>java Test
Enter number of rotations(r):
4
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[2, 3, 4, 5, 1]

C:\prakashclasses>java Test
Enter number of rotations(r):
5
Before Rotation==>[1, 2, 3, 4, 5]
After Rotation ==>[1, 2, 3, 4, 5]
```

```
Method1 ----> Brute Force
Method2 ----> by using temp variable
Method3 ----> by using temp array ---> m1
Method4 ----> by using temp array ---> m2
Method5 ----> by using reversal algo
```

Introduction to Algorithms
Recursion
Arrays
Matrices
Array Rotataions
Searching and Sorting Tech.

Sorting:-

Arranging the data in asc order or desc order is called as sorting. there various sorting tech are existed.

bubble sort:

Ex:

```java
import java.util.*;

class Demo
{
	static void bubbleSortAsc(int a[])
	{
		int i,j,t;
		for(i=0;i<a.length-1;i++)
		{
			//bubble sort
			for(j=0;j<a.length-i-1;j++)
			{
```

```
                        if(a[j] > a[j+1])
                        {
                              t = a[j];
                              a[j] = a[j+1];
                              a[j+1] = t;
                        }
                  }
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {1,9,3,8,7,5,2,4};
            System.out.println("before
sorting=====>"+Arrays.toString(a));
            Demo.bubbleSortAsc(a);
            System.out.println("after sorting
asc==>"+Arrays.toString(a));
      }
}

output:
-------
before sorting=====>[1, 9, 3, 8, 7, 5, 2, 4]
after sorting asc==>[1, 2, 3, 4, 5, 7, 8, 9]

Ex:
---
import java.util.*;

class Demo
{
      static void bubbleSortDesc(int a[])
      {
            int i,j,t;
            for(i=0;i<a.length-1;i++)
            {
                  //bubble sort
                  for(j=0;j<a.length-i-1;j++)
                  {
                        if(a[j] < a[j+1])
                        {
                              t = a[j];
                              a[j] = a[j+1];
                              a[j+1] = t;
                        }
                  }
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {1,9,3,8,7,5,2,4};
            System.out.println("before 
sorting=====>"+Arrays.toString(a));
            Demo.bubbleSortDesc(a);
            System.out.println("after sorting 
desc=>"+Arrays.toString(a));
      }
}


output:
-------
before sorting=====>[1, 9, 3, 8, 7, 5, 2, 4]
after sorting desc=>[9, 8, 7, 5, 4, 3, 2, 1]

selection sort:
~~~~~~~~~~~~~~~
Ex:
---
import java.util.*;

class Demo
{
      static void selectionSortAsc(int a[])
      {
            int i,j,min,temp,n=a.length;

            for(i=0;i<n-1;i++)
            {
                  min = i;
                  for(j=i+1;j<n;j++)
                  {
                        if(a[j]<a[min])
                        {
                              min = j;
                        }
                  }
                  if(min!=i)
                  {
                        temp = a[i];
                        a[i] = a[min];
                        a[min] = temp;
                  }
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {1,9,3,8,7,5,2,4};
```

```
            System.out.println("before
sorting=====>"+Arrays.toString(a));
            Demo.selectionSortAsc(a);
            System.out.println("after sorting asc=>"+Arrays.toString(a));
      }
}
```

```
output:
-------
before sorting=====>[1, 9, 3, 8, 7, 5, 2, 4]
after sorting asc=>[1, 2, 3, 4, 5, 7, 8, 9]
```

```
Ex:
---
import java.util.*;

class Demo
{
      static void selectionSortDesc(int a[])
      {
            int i,j,min,temp,n=a.length;

            for(i=0;i<n-1;i++)
            {
                  min = i;
                  for(j=i+1;j<n;j++)
                  {
                        if(a[j]>a[min])
                        {
                              min = j;
                        }
                  }
                  if(min!=i)
                  {
                        temp = a[i];
                        a[i] = a[min];
                        a[min] = temp;
                  }
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            int a[] = {1,9,3,8,7,5,2,4};
            System.out.println("before
sorting=====>"+Arrays.toString(a));
            Demo.selectionSortDesc(a);
            System.out.println("after sorting
desc=>"+Arrays.toString(a));
      }
}
```

output:

```
before sorting=====>[1, 9, 3, 8, 7, 5, 2, 4]
after sorting desc=>[9, 8, 7, 5, 4, 3, 2, 1]
```

insertion sort:

Ex:

```
import java.util.*;

class Demo
{
      static void insertionSortAsc(int a[])
      {
            int i,j,temp,n=a.length;

            for(i=1;i<n;i++)
            {
                  temp = a[i];
                  j=i-1;
                  while(j>=0 && a[j]>temp)
                  {
                        a[j+1] = a[j];
                        j--;
                  }
                  a[j+1] = temp;
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int a[] = new int[10];
            for(int i=0;i<a.length;i++)
                  a[i] = r.nextInt(100);
            System.out.println("before
sorting=====>"+Arrays.toString(a));
            Demo.insertionSortAsc(a);
            System.out.println("after sorting
desc=>"+Arrays.toString(a));
      }
}
```

output:

```
before sorting=====>[46, 17, 54, 88, 93, 2, 7, 7, 60, 9]
after sorting desc=>[2, 7, 7, 9, 17, 46, 54, 60, 88, 93]

before sorting=====>[31, 15, 77, 93, 68, 9, 78, 69, 23, 11]
after sorting desc=>[9, 11, 15, 23, 31, 68, 69, 77, 78, 93]
```

```
Ex:
---
import java.util.*;

class Demo
{
      static void insertionSortDesc(int a[])
      {
            int i,j,temp,n=a.length;

            for(i=1;i<n;i++)
            {
                  temp = a[i];
                  j=i-1;
                  while(j>=0 && a[j]<temp)
                  {
                        a[j+1] = a[j];
                        j--;
                  }
                  a[j+1] = temp;
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int a[] = new int[10];
            for(int i=0;i<a.length;i++)
                  a[i] = r.nextInt(100);
            System.out.println("before
sorting=====>"+Arrays.toString(a));
            Demo.insertionSortDesc(a);
            System.out.println("after sorting
desc=>"+Arrays.toString(a));
      }
}

output:
-------
before sorting=====>[62, 85, 35, 18, 65, 82, 5, 21, 33, 92]
after sorting desc=>[92, 85, 82, 65, 62, 35, 33, 21, 18, 5]

Quick Sort:-
~~~~~~~~~~~
Ex:
---
import java.util.*;

class Demo
{
```

```
    static void quickSortAsc(int a[],int lIndex,int hIndex){
        if(lIndex>=hIndex) //terminate recursion or base condition
            return;
        int pivot, lp, rp, temp;

        pivot = a[hIndex];
        lp = lIndex;
        rp = hIndex;

        while(lp<rp){
            while(a[lp]<=pivot && lp<rp)
                lp++;
            while(a[rp]>=pivot && lp<rp)
                rp--;
            temp = a[lp];
            a[lp] = a[rp];
            a[rp] = temp;
        }

        temp = a[lp];
        a[lp]=a[hIndex];
        a[hIndex]=temp;

        quickSortAsc(a,lIndex,lp-1);
        quickSortAsc(a,lp+1,hIndex);
    }
}
class Test
{
    public static void main(String[] args)
    {
        Random r = new Random();
        int a[] = new int[10];
        for(int i=0;i<a.length;i++)
            a[i] = r.nextInt(100);
        System.out.println("before 
sorting=====>"+Arrays.toString(a));
        Demo.quickSortAsc(a,0,a.length-1);
        System.out.println("after sorting asc=>"+Arrays.toString(a));
    }
}

output:
-------
before sorting=====>[44, 74, 13, 41, 56, 39, 91, 68, 25, 60]
after sorting asc=>[13, 25, 39, 41, 44, 56, 60, 68, 74, 91]


Ex:
---
import java.util.*;

class Demo
{
      static void quickSortDesc(int a[],int lIndex,int hIndex){
            if(lIndex>=hIndex) //terminate recursion or base condition
                  return;
            int pivot, lp, rp, temp;

            pivot = a[hIndex];
            lp = lIndex;
            rp = hIndex;

            while(lp<rp){
                  while(a[lp]>=pivot && lp<rp)
                        lp++;
                  while(a[rp]<=pivot && lp<rp)
                        rp--;
                  temp = a[lp];
                  a[lp] = a[rp];
                  a[rp] = temp;
            }

            temp = a[lp];
            a[lp]=a[hIndex];
            a[hIndex]=temp;

            quickSortDesc(a,lIndex,lp-1);
            quickSortDesc(a,lp+1,hIndex);
      }
}
class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int a[] = new int[10];
            for(int i=0;i<a.length;i++)
                  a[i] = r.nextInt(100);
            System.out.println("before
sorting=====>"+Arrays.toString(a));
            Demo.quickSortDesc(a,0,a.length-1);
            System.out.println("after sorting
desc=>"+Arrays.toString(a));
      }
}
```

```
output:
-------
before sorting=====>[27, 61, 26, 63, 57, 36, 14, 20, 33, 40]
after sorting desc=>[63, 61, 57, 40, 36, 33, 27, 26, 20, 14]
```

merge sort:
----

divide and combine

```
Ex:
---
import java.util.*;

class Demo
{
      static void mergeSort(int[] a,int n)
      {
            if(n<2) //base condition
                  return;
            int mid=n/2;
            int l[] = new int[mid];
            int r[] = new int[n-mid];
            int i;
            for(i=0;i<mid;i++)
                  l[i]=a[i];
            for(i=mid;i<n;i++)
                  r[i-mid]=a[i];
            mergeSort(l,mid);
            mergeSort(r,n-mid);
            merge(a,l,r,mid,n-mid);
      }
      static void merge(int a[],int l[],int r[],int left,int right){
            int i=0,j=0,k=0;
            while(i<left && j<right){
                  if(l[i]<=r[j])
                        a[k++]=l[i++];
                  else
                        a[k++]=r[j++];
            }
            while(i<left)
                  a[k++]=l[i++];
            while(j<right)
                  a[k++]=r[j++];
      }
}

class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int[] a = new int[10];

            for(int i=0;i<a.length;i++)
            {
                  a[i] = r.nextInt(100);
            }

            System.out.println("Before Sorting====>
"+Arrays.toString(a));
            Demo.mergeSort(a,a.length);
            System.out.println("After Sorting====> "+Arrays.toString(a));
      }
}
```

```
output:
-------
Before Sorting====> [61, 36, 17, 78, 23, 36, 58, 47, 11, 9]
After Sorting====> [9, 11, 17, 23, 36, 36, 47, 58, 61, 78]
```

Ex:

```
import java.util.*;

class Demo
{
      static void mergeSort(int[] a,int n)
      {
            if(n<2) //base condition
                  return;
            int mid=n/2;
            int l[] = new int[mid];
            int r[] = new int[n-mid];
            int i;
            for(i=0;i<mid;i++)
                  l[i]=a[i];
            for(i=mid;i<n;i++)
                  r[i-mid]=a[i];
            mergeSort(l,mid);
            mergeSort(r,n-mid);
            merge(a,l,r,mid,n-mid);
      }
      static void merge(int a[],int l[],int r[],int left,int right){
            int i=0,j=0,k=0;
            while(i<left && j<right){
                  if(l[i]>=r[j])
                        a[k++]=l[i++];
                  else
                        a[k++]=r[j++];
            }
            while(i<left)
                  a[k++]=l[i++];
            while(j<right)
                  a[k++]=r[j++];
      }
}

class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int[] a = new int[10];

            for(int i=0;i<a.length;i++)
            {
                  a[i] = r.nextInt(100);
            }

            System.out.println("Before Sorting====> 
"+Arrays.toString(a));
            Demo.mergeSort(a,a.length);
            System.out.println("After Sorting====> "+Arrays.toString(a));
      }
}
```

```
output:
-------
Before Sorting====> [14, 82, 65, 12, 60, 44, 80, 96, 52, 35]
After Sorting====> [96, 82, 80, 65, 60, 52, 44, 35, 14, 12]
```

shell sorting:

Ex:

```
import java.util.*;

class Demo
{
      static void shellSortAsc(int[] a,int n)
      {
            int gap,i,j,temp;
            for(gap=n/2;gap>=1;gap=gap/2)
            {
                  for(j=gap;j<n;j++)
                  {
                        for(i=j-gap;i>=0;i=i-gap)
                        {
                              if(a[i+gap]>a[i])
                                    break;
                              else
                              {
                                    temp=a[i+gap];
                                    a[i+gap]=a[i];
                                    a[i]=temp;
                              }
                        }
                  }
            }
      }
}

class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int[] a = new int[10];
            for(int i=0;i<a.length;i++)
            {
                  a[i] = r.nextInt(100);
            }

            System.out.println("Before Sorting====> 
"+Arrays.toString(a));
            Demo.shellSortAsc(a,a.length);
            System.out.println("After Sorting====> "+Arrays.toString(a));
      }
}
```

```
output:
-------
Before Sorting====> [5, 9, 68, 8, 60, 7, 89, 31, 35, 15]
After Sorting====> [5, 7, 8, 9, 15, 31, 35, 60, 68, 89]
```

```
Ex:
---
import java.util.*;

class Demo
{
      static void shellSortDesc(int[] a,int n)
      {
            int gap,i,j,temp;
            for(gap=n/2;gap>=1;gap=gap/2)
            {
                  for(j=gap;j<n;j++)
                  {
                        for(i=j-gap;i>=0;i=i-gap)
                        {
                              if(a[i+gap]<a[i])
                                    break;
                              else
                              {
                                    temp=a[i+gap];
                                    a[i+gap]=a[i];
                                    a[i]=temp;
                              }
                        }
                  }
            }
      }
}

class Test
{
      public static void main(String[] args)
      {
            Random r = new Random();
            int[] a = new int[10];
            for(int i=0;i<a.length;i++)
            {
                  a[i] = r.nextInt(100);
            }

            System.out.println("Before Sorting====>
"+Arrays.toString(a));
            Demo.shellSortDesc(a,a.length);
            System.out.println("After Sorting====> "+Arrays.toString(a));
      }
}
```

```
output:
-------
Before Sorting====> [86, 70, 36, 98, 4, 59, 58, 41, 44, 14]
After Sorting====> [98, 86, 70, 59, 58, 44, 41, 36, 14, 4]
```

searching algo:

it is used to check whether an obj is existed in the array or not.

Linear and Binary search

Ex:

```
import java.util.*;

class Demo
{
      static int linearSearch(int a[],int key){
            int i,index=-1;
            for(i=0;i<a.length;i++){
                  if(key==a[i])
                  {
                        index=i;
                        break;
                  }
            }
            return index;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int[] a = {10, 11, 12, 13, 11, 12, 11, 8, 19, 11};
            System.out.println("Array="+Arrays.toString(a));
            System.out.println("Enter key element to search:");
            int key = obj.nextInt();

            System.out.println(Demo.linearSearch(a,key));
      }
```

```
}

output:
-------
C:\prakashclasses>java Test
Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
Enter key element to search:
8
7

C:\prakashclasses>java Test
Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
Enter key element to search:
11
1

C:\prakashclasses>java Test
Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
Enter key element to search:
99
-1

Ex:
---
import java.util.*;

class Demo
{
      static ArrayList linearSearch(int a[],int key){
            int i,c=0;
            ArrayList list = new ArrayList();
            for(i=0;i<a.length;i++){
                  if(key==a[i])
                  {
                        list.add(i);
                        c++;
                        if(c>=2)
                              break;

                  }
            }
            return list;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int[] a = {10, 11, 12, 13, 11, 12, 11, 8, 19, 11};
            System.out.println("Array="+Arrays.toString(a));
            System.out.println("Enter key element to search:");
            int key = obj.nextInt();
```

```
                  System.out.println(Demo.linearSearch(a,key));
            }
      }

      output:
      -------
      C:\prakashclasses>java Test
      Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
      Enter key element to search:
      11
      [1, 4]

      C:\prakashclasses>java Test
      Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
      Enter key element to search:
      10
      [0]

      C:\prakashclasses>java Test
      Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
      Enter key element to search:
      12
      [2, 5]

      Ex:
      ---
      import java.util.*;

      class Demo
      {
            static ArrayList linearSearch(int a[],int key){
                  int i;
                  ArrayList list = new ArrayList();
                  for(i=0;i<a.length;i++){
                        if(key==a[i])
                              list.add(i);
                  }
                  return list;
            }
      }

      class Test
      {
            public static void main(String[] args)
            {
                  Scanner obj = new Scanner(System.in);
                  int[] a = {10, 11, 12, 13, 11, 12, 11, 8, 19, 11};
                  System.out.println("Array="+Arrays.toString(a));
                  System.out.println("Enter key element to search:");
                  int key = obj.nextInt();

                  System.out.println(Demo.linearSearch(a,key));
            }
```

```
}

output:
-------
C:\prakashclasses>java Test
Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
Enter key element to search:
10
[0]

C:\prakashclasses>java Test
Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
Enter key element to search:
12
[2, 5]

C:\prakashclasses>java Test
Array=[10, 11, 12, 13, 11, 12, 11, 8, 19, 11]
Enter key element to search:
11
[1, 4, 6, 9]

Ex:
---
import java.util.*;

class Demo
{
      static int binarySearch(int a[],int key){
            int l=0,h=a.length-1,mid;
            while(l<=h){
                  mid=(l+h)/2;
                  if(a[mid]==key)
                        return mid;
                  else if(key<a[mid])
                        h=mid-1;
                  else
                        l=mid+1;
            }
            return -1;
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int[] a = {10, 34, 23, 22, 56, 65, 77, 78, 87, 99};
            Arrays.sort(a);
            System.out.println("Array="+Arrays.toString(a));
            System.out.println("Enter key element to search:");
            int key = obj.nextInt();
            System.out.println(Demo.binarySearch(a,key));
      }
}
```

```
output:
-------
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
Array=[10, 22, 23, 34, 56, 65, 77, 78, 87, 99]
Enter key element to search:
34
3

C:\prakashclasses>java Test
Array=[10, 22, 23, 34, 56, 65, 77, 78, 87, 99]
Enter key element to search:
88
-1
```

```
Ex:
---
import java.util.*;

class Demo
{
      static int binarySearch(int a[],int key,int l,int h){
            int mid=(l+h)/2;
            if(l>h)
                  return -1;
            if(key==a[mid])
                  return mid;
            else if(key<a[mid])
                  return binarySearch(a,key,l,mid-1);
            else
                  return binarySearch(a,key,mid+1,h);
      }
}

class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            int[] a = {10, 34, 23, 22, 56, 65, 77, 78, 87, 99};
            Arrays.sort(a);
            System.out.println("Array="+Arrays.toString(a));
            System.out.println("Enter key element to search:");
            int key = obj.nextInt();

            System.out.println(Demo.binarySearch(a,key,0,a.length-1));
      }
}
```

output:

```
C:\prakashclasses>java Test
Array=[10, 22, 23, 34, 56, 65, 77, 78, 87, 99]
Enter key element to search:
88
-1

C:\prakashclasses>java Test
Array=[10, 22, 23, 34, 56, 65, 77, 78, 87, 99]
Enter key element to search:
23
2

C:\prakashclasses>java Test
Array=[10, 22, 23, 34, 56, 65, 77, 78, 87, 99]
Enter key element to search:
77
6

C:\prakashclasses>java Test
Array=[10, 22, 23, 34, 56, 65, 77, 78, 87, 99]
Enter key element to search:
99
9
```

searching and sorting techniques

01. Introduction to searching and sorting
02. Sample implementations for searching and sorting
03. Arrays.sort() method
04. Random values generation for an array
05. Bubble sort
06. Selection Sort
07. Insertion sort
08. Quick sort
09. Merge sort
10. Shell sort
11. Linear search (3cases)
12. Binary searching using iteration and recursion.
13. Sample programs
14. Time complexities

Linked List:

=> The problems with arrays are
1) fixed in size
2) continoues memory locations

=> we can keep the data in any where, those data's are linked with a link. so that we can perform all our operations very effectively.

```
=> types of LL
                        1) single linked list
                        2) double linked list
                        3) circular single linked list
                        4) circular double linked list
```

Representation of LL:

Every linked list contains two parts, data and next elements, data is used to represent content and next is used to point to next data reference.

diagram

Ex:
---
      adding an node at begining, ending, deleting node from the begin, end, printing

Ex:

```
class LL
{
      Node head;
      class Node
      {
            int data;
            Node next;
            Node(int data){
                  this.data = data;
                  this.next = null;
            }
      }
      void addFirst(int data){
            Node newNode = new Node(data);
            if(head==null){
                  head=newNode;
                  return;
            }
            newNode.next = head;
            head = newNode;
      }
      void addLast(int data){
            Node newNode = new Node(data);
            if(head==null){
                  head=newNode;
                  return;
            }
            Node temp = head;
            while(temp.next!=null)
                  temp = temp.next;
            temp.next=newNode;
      }
      void printList(){
            if(head==null)
            {
                  System.out.println("list is empty");
                  return;
            }
            Node temp = head;
            while(temp!=null)
            {
                  System.out.print(temp.data+" => ");
                  temp = temp.next;
            }
            System.out.println("NULL");
      }
      void deleteFirst(){
            if(head==null){
                  System.out.println("List is empty");
                  return;
            }
            head = head.next;
      }
      void deleteLast(){
            if(head==null)
            {
                  System.out.println("list is empty");
                  return;
            }
            if(head.next==null){
                  head=null;
                  return;
            }
            Node temp1,temp2;
            temp1=head;
            temp2=head.next;
            while(temp2.next!=null){
                  temp2 = temp2.next;
                  temp1 = temp1.next;
            }
            temp1.next = null;
      }
}

class Test
{
      public static void main(String[] args)
      {
            LL list = new LL();
            list.addFirst(333);
            list.addFirst(222);
            list.addFirst(111);
            list.addLast(444);
            list.addLast(555);
            list.addLast(666);
            list.printList();//111=>222=>333=>444=>555=>666=>NULL
            list.deleteFirst();
            list.deleteFirst();
            list.printList();//333=>444=>555=>666=>NULL
            list.deleteLast();
            list.deleteLast();
            list.printList();//333=>444=>NULL
      }
}
```

```
output:
-------
111 => 222 => 333 => 444 => 555 => 666 => NULL
333 => 444 => 555 => 666 => NULL
333 => 444 => NULL
```

The following are the operations that we can perform on linked list

01) Inserting the data first
02) Inserting the data last
03) Inserting the data at position
04) Sorted Insertion Asc/Desc
05) Traversing or Displaying
06) Size or Length of list
07) Reverse List
08) Searching
09) Deleting from first
10) Deleting from last
11) Delete from position
12) Deleting Element
13) Deleting Elements
14) Deleting Duplicates
15) Copy the reversed list
16) Copy of original list
17) Comparing two list objects
18) Finding nth node from begining and ending

```
Single Linked List Implementation
---------------------------------
01) Inserting the data first  ------------> ok
02) Inserting the data last --------------> ok
03) Inserting the data at position -------> ok
04) Sorted Insertion Asc/Desc ------------> ok
05) Traversing or Displaying -------------> ok
06) Size or Length of list ---------------> ok
07) Reverse List -------------------------> ok
08) Searching ----------------------------> ok
09) Deleting from first ------------------> ok
10) Deleting from last -------------------> ok
11) Delete from position -----------------> ok
12) Deleting Element ---------------------> ok
13) Deleting Elements --------------------> ok
```

14) Deleting Duplicates ------------------> ok
15) Copy the reversed list ---------------> ok
16) Copy of original list ----------------> ok
17) Comparing two list objects ------------------> ok
18) Finding nth node from begining and ending ---> ok

Ex:

```
import java.util.*;

class SLL{
      Node head;
      int size;
      class Node{
            int data;
            Node next;
            Node(int data){
                  this.data = data;
                  this.next = null;
                  size++;
            }
            Node(int data,Node temp){
                  this.data = data;
                  this.next = temp;
                  size++;
            }
      }
      int getSize(){
            return this.size;
      }
      void addFirst(int data){
            Node newNode = new Node(data);
            if(head==null){
                  head = newNode;
                  return;
            }
            newNode.next = head;
            head = newNode;
      }
      void addLast(int data){
            Node newNode = new Node(data);
            if(head==null){
                  head = newNode;
                  return;
            }
            Node currNode = head;
            while(currNode.next!=null)
                  currNode = currNode.next;
            currNode.next = newNode;
      }
      void addPos(int data,int pos){
            int i=0;
            Node newNode = new Node(data);
            if(head==null){
                  head = newNode;
                  return;
            }
            if(pos!=0){
                  Node currNode = head;
                  Node prevNode = null;
                  while(currNode.next!=null && i<pos){
                        prevNode = currNode;
                        currNode = currNode.next;
                        i++;
                  }
                  prevNode.next = newNode;
                  newNode.next = currNode;
            }
            else{
                  newNode.next = head;
                  head = newNode;
            }
      }
      void sortedInsertAsc(int data){
            Node newNode = new Node(data);
            Node currNode = head;
            if(currNode==null||currNode.data>data){
                  newNode.next = head;
                  head = newNode;
                  return;
            }
            while(currNode.next!=null && currNode.next.data<data){
                  currNode = currNode.next;
            }
            newNode.next = currNode.next;
            currNode.next = newNode;
      }
      void sortedInsertDesc(int data){
            Node newNode = new Node(data);
            Node currNode = head;
            if(currNode==null||currNode.data<data){
                  newNode.next = head;
                  head = newNode;
                  return;
            }
            while(currNode.next!=null && currNode.next.data>data){
                  currNode = currNode.next;
            }
            newNode.next = currNode.next;
            currNode.next = newNode;
      }
      void deleteFirst(){
            if(head==null){
                  System.out.println("List is empty");
                  return;
            }
            size--;
            head=head.next;
```

```
}
void deleteLast(){
      if(head==null){
            System.out.println("list is empty");
            return;
      }
      if(head.next==null){
            head=null;
            return;
      }
      size--;
      Node temp1=head,temp2=head.next;
      while(temp2.next!=null){
            temp2 = temp2.next;
            temp1 = temp1.next;
      }
      temp1.next = null;
}
void deleteElement(int data){
      Node temp = head;
      if(temp==null){
            System.out.println("empty");
            return;
      }
      if(temp.data == data){
            head = head.next;
            size--;
            return;
      }
      while(temp.next!=null){
            if(temp.next.data == data){
                  temp.next = temp.next.next;
                  size--;
                  return;
            }
            temp = temp.next;
      }
}
void deleteElements(int data){
      Node temp = head;
      if(temp==null){
            System.out.println("empty");
            return;
      }
      if(temp.data == data){
            head = head.next;
            size--;
      }
      while(temp.next!=null){
            if(temp.next.data == data){
                  temp.next = temp.next.next;
                  size--;
            }
            if(temp.next!=null)
            temp = temp.next;
    }
}
void deleteElementAtPos(int pos){
    Node temp = head;
    int i=0;
    if(temp==null){
        System.out.println("empty");
        return;
    }
    if(pos==0){
        head = head.next;
        size--;
        return;
    }
    while(temp.next!=null && i<pos){
        if(i==pos-1){
            temp.next = temp.next.next;
            size--;
            return;
        }
        i++;
        temp = temp.next;
    }
}
void printList(){
    if(head==null){
        System.out.println("list is empty");
    }
    Node currNode = head;
    while(currNode!=null){
        System.out.print(currNode.data+" => ");
        currNode = currNode.next;
    }
    System.out.println("null");
}
boolean search(int data){
    Node currNode = head;
    while(currNode!=null){
        if(currNode.data == data)
            return true;
        currNode = currNode.next;
    }
    return false;
}
void reverse(){
    Node curr = head, prev=null,next=null;
    while(curr!=null){
        next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;
    }
    head = prev;
```

```
      }
      void reverseR(){
            head = reverseRUtil(head,null);
      }
      Node reverseRUtil(Node currNode, Node nextNode){
            Node res;
            if(currNode==null)
                  return null;
            if(currNode.next==null){
                  currNode.next = nextNode;
                  return currNode;
            }
            res = reverseRUtil(currNode.next,currNode);
            currNode.next = nextNode;
            return res;
      }
      void removeDuplicates(){
            Node currNode = head;
            while(currNode!=null){
                  if(currNode.next!=null && currNode.data ==
currNode.next.data)
                        currNode.next = currNode.next.next;
                  else
                        currNode = currNode.next;
            }
      }
      SLL copyReversedList(){
            Node temp1=null,temp2=null,currNode=head;
            while(currNode!=null){
                  temp2 = new Node(currNode.data,temp1);
                  currNode = currNode.next;
                  temp1 = temp2;
            }
            SLL obj = new SLL();
            obj.head = temp1;
            return obj;
      }
      SLL copyList(){
            Node headNode=null,tailNode=null,tempNode=null,currNode=head;
            if(currNode==null)
                  return null;
            headNode = new Node(currNode.data,null);
            tailNode = headNode;
            currNode = currNode.next;
            while(currNode!=null){
                  tempNode = new Node(currNode.data,null);
                  tailNode.next = tempNode;
                  tailNode = tempNode;
                  currNode = currNode.next;
            }
            SLL obj = new SLL();
            obj.head = headNode;
            return obj;
      }
      boolean compareList1(SLL list){
            Node head1=head,head2=list.head;
            while(head1!=null && head2!=null){
                  if(head1.data!=head2.data)
                        return false;
                  head1=head1.next;
                  head2=head2.next;
            }
            if(head1==null && head2==null)
                  return true;
            return false;
      }
      boolean compareList2(SLL list){
            return compareList(head,list.head);
      }
      boolean compareList(Node head1,Node head2){
            if(head1==null && head2==null)
                  return true;
            else if(head1==null || head2==null ||
(head1.data!=head2.data))
                  return false;
            else
                  return compareList(head1.next,head2.next);
      }
      int nthNodeFromBegin(int index){
            if(index>getSize() || index<1)
                  return -1;
            int count=0;
            Node currNode = head;
            while(currNode!=null && count<index-1){
                  count++;
                  currNode=currNode.next;
            }
            return currNode.data;
      }
      int nthNodeFromEnd(int index){
            int size = getSize();
            int sindex;
            if(size!=0 && size<index)
                  return -1;
            sindex = size-index+1;
            return nthNodeFromBegin(sindex);
      }
}

class Test
{
      public static void main(String[] args)
      {
            SLL list1 = new SLL();
            list1.addLast(111);
            list1.addLast(222);
            list1.addLast(333);
            list1.addLast(444);
            list1.addLast(555);
            list1.addLast(666);
            list1.addLast(777);
            list1.addLast(888);
            list1.printList();
            System.out.println(list1.nthNodeFromBegin(3));
            System.out.println(list1.nthNodeFromEnd(3));
      }
}
```

```
Double Linked List
------------------
Double Linked List Implementation
---------------------------------
01) Inserting the data first         ------------> ok
02) Inserting the data last        --------------> ok
03) Inserting the data at position        -------> ok
04) Sorted Insertion Asc/Desc        ------------> ok
05) Traversing or Displaying        -------------> ok
06) Size or Length of list        ---------------> ok
07) Reverse List          -----------------------> ok
08) Searching        ----------------------------> ok
09) Deleting from first           -----------------> ok
10) Deleting from last        -------------------> ok
11) Delete from position        -----------------> ok
12) Deleting Element        ---------------------> ok
13) Deleting Elements        --------------------> ok
14) Deleting Duplicates        ------------------> ok
15) Copy the reversed list        ---------------> ok
16) Copy of original list         ---------------> ok
17) Comparing two list objects ------------------> ok
18) Finding nth node from begining and ending ---> ok
```

Ex:

```
public class DLL
{
      Node head;
      int size = 0;
      class Node{
            int data;
            Node next,prev;
            Node(int data,Node next,Node prev){
                  this.data = data;
                  this.next = next;
                  this.prev = prev;
                  size++;
            }
      }
      void traverse() {
            if(head==null) {
                  System.out.println("List is Empty");
                  return;
```

```
        }
        Node currNode = head;
        while(currNode!=null) {
            System.out.print(currNode.data+" => ");
            currNode = currNode.next;
        }
        System.out.println("NULL");
    }
    void addFirst(int data) {
        Node newNode = new Node(data,null,null);
        if(head==null)
            head = newNode;
        else {
            head.prev = newNode;
            newNode.next = head;
            head = newNode;
        }
    }
    void addLast(int data) {
        Node newNode = new Node(data,null,null);
        if(head==null)
            head = newNode;
        else {
            Node currNode = head;
            while(currNode.next != null)
                currNode = currNode.next;
            currNode.next = newNode;
            newNode.prev = currNode;
        }
    }
    void addPos(int data,int pos) {
        int i=0;
        if(pos<0 || pos>=size) {
            System.out.println("out of range");
            return;
        }
        Node newNode = new Node(data,null,null);
        if(head==null) {
            head = newNode;
            return;
        }
        if(pos!=0) {
            Node currNode = head, temp = null;
            while(currNode.next!=null && i<pos) {
                temp=currNode;
                currNode = currNode.next;
                i++;
            }
            temp.next = newNode;
            newNode.prev = temp;
            newNode.next = currNode;
            currNode.prev = newNode;
        }
        else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
      }
}
void sortedInsertAsc(int data) {
      Node newNode = new Node(data,null,null);
      Node currNode = head;
      if(currNode==null) {
            head = newNode;
            return;
      }
      if(currNode.data>data) {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
            return;
      }

      while(currNode.next!=null && currNode.next.data < data)
            currNode = currNode.next;

      if(currNode.next!=null) {
            newNode.next = currNode.next;
            currNode.next.prev = newNode;
            currNode.next = newNode;
            newNode.prev = currNode;
      }
      else {
            currNode.next = newNode;
            newNode.prev = currNode;
      }
}
void sortedInsertDesc(int data) {
      Node newNode = new Node(data,null,null);
      Node currNode = head;
      if(currNode==null) {
            head = newNode;
            return;
      }
      if(currNode.data<data) {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
            return;
      }

      while(currNode.next!=null && currNode.next.data > data)
            currNode = currNode.next;

      if(currNode.next!=null) {
            newNode.next = currNode.next;
            currNode.next.prev = newNode;
            currNode.next = newNode;
```

```
        newNode.prev = currNode;
    }
    else {
        currNode.next = newNode;
        newNode.prev = currNode;
    }
}
int getSize() {
    return this.size;
}
boolean search(int data) {
    Node temp = head;
    while(temp!=null) {
        if(temp.data == data)
            return true;
        temp = temp.next;
    }
    return false;
}
void deleteFirst() {
    if(head==null) {
        System.out.println("DLL is empty");
        return;
    }
    size--;
    head = head.next;
    if(head!=null)
        head.prev = null;
}
void deleteLast() {
    if(head==null) {
        System.out.println("DLL is empty");
        return;
    }
    if(head.next == null) {
        head = null;
        size--;
        return;
    }
    size--;
    Node temp1 = head, temp2 = head.next;
    while(temp2.next!=null) {
        temp2 = temp2.next;
        temp1 = temp1.next;
    }
    temp1.next = null;
}
void deleteElementAtPos(int pos) {
    Node temp1 = head, temp2;
    int i=0;
    if(temp1==null) {
        System.out.println("DLL is empty");
        return;
    }
```

```
      if(pos<0 || pos>=size) {
            System.out.println("out of range");
            return;
      }
      if(pos==0) {
            head = head.next;
            if(head!=null)
                  head.prev = null;
            size--;
            return;
      }
      while(temp1.next!=null && i<pos) {
            if(i==pos-1) {
                  temp1.next = temp1.next.next;
                  temp2 = temp1.next;
                  if(temp2!=null)
                        temp2.prev = temp1;
                  size--;
                  return;
            }
            i++;
            temp1 = temp1.next;
            temp2 = temp1.next;
      }
}
void deleteElement(int data) {
      Node temp1 = head, temp2;
      if(temp1==null) {
            System.out.println("DLL empty");
            return;
      }
      if(temp1.data == data) {
            head = head.next;
            if(head!=null)
                  head.prev = null;
            size--;
            return;
      }
      while(temp1.next!=null)
      {
            if(temp1.next.data == data) {
                  temp1.next = temp1.next.next;
                  temp2 = temp1.next;
                  if(temp2!=null)
                        temp2.prev = temp1;
                  size--;
                  return;
            }
            temp1 = temp1.next;
      }

}
void deleteElements(int data) {
      Node temp1 = head, temp2;
```

```
            if(temp1==null) {
                  System.out.println("DLL empty");
                  return;
            }
            if(temp1.data == data) {
                  head = head.next;
                  if(head!=null)
                        head.prev = null;
                  size--;
            }
            while(temp1.next!=null)
            {
                  if(temp1.next.data == data) {
                        temp1.next = temp1.next.next;
                        temp2 = temp1.next;
                        if(temp2!=null)
                              temp2.prev = temp1;
                        size--;
                  }
                  if(temp1.next!=null)
                        temp1 = temp1.next;
            }
      }
      void removeDuplicates() {
            Node currNode = head, temp;
            while(currNode!=null) {
                  if(currNode.next!=null && currNode.data ==
currNode.next.data) {
                        currNode.next = currNode.next.next;
                        temp = currNode.next;
                        if(temp!=null)
                              temp.prev = currNode;
                  }
                  else
                        currNode = currNode.next;
            }
      }
      DLL copyList() {
            Node headNode=null,tailNode=null,tempNode=null,currNode=head;
            if(currNode==null)
                  return null;
            headNode = new Node(currNode.data,null,null);
            tailNode = headNode;
            currNode = currNode.next;
            while(currNode!=null) {
                  tempNode = new Node(currNode.data,null,null);
                  tailNode.next = tempNode;
                  tempNode.prev = tailNode;
                  tailNode = tempNode;
                  currNode = currNode.next;
            }
            DLL obj = new DLL();
            obj.head = headNode;
            return obj;
      }
      DLL copyReversedList() {
            Node temp1=null,temp2=null,currNode=head;
            while(currNode!=null) {
                  temp2=new Node(currNode.data,temp1,null);
                  currNode = currNode.next;
                  if(temp1!=null)
                        temp1.prev = temp2;
                  temp1 = temp2;
            }
            DLL obj = new DLL();
            obj.head = temp1;
            return obj;
      }
      boolean compareListI(DLL list) {
            Node head1=head,head2=list.head;
            while(head1!=null && head2!=null) {
                  if(head1.data!=head2.data)
                        return false;
                  head1 = head1.next;
                  head2 = head2.next;
            }
            if(head1==null && head2==null)
                  return true;
            return false;
      }
      boolean compareListR(DLL list) {
            return compareList(head,list.head);
      }
      boolean compareList(Node head1,Node head2) {
            if(head1==null && head2==null)
                  return true;
            else if(head1==null || head2==null ||
(head1.data!=head2.data))
                  return false;
            else
                  return compareList(head1.next,head2.next);
      }
      int nthNodeFromBegin(int index) {
            if(index>getSize() || index<1)
                  return -1;
            int count=0;
            Node currNode = head;
            while(currNode!=null && count<index-1) {
                  count++;
                  currNode = currNode.next;
            }
            return currNode.data;
      }
      int nthNodeFromEnd(int index) {
            int size = getSize();
            int sindex;
            if(size!=0 && size<index)
                  return -1;
            sindex = size-index+1;
            return nthNodeFromBegin(sindex);
      }
      void reverse() {
            Node temp=null,currNode=head;
            while(currNode!=null) {
                  temp = currNode.prev;
                  currNode.prev = currNode.next;
                  currNode.next = temp;
                  currNode= currNode.prev;
            }
            if(temp!=null)
                  head = temp.prev;
      }
}

circular single linked list
---------------------------
public class CSL {
      Node tail;
      int size = 0;
      class Node{
            int value;
            Node next;
            Node(int value,Node next){
                  this.value = value;
                  this.next = next;
            }
      }
      void print() {
            if(size==0) {
                  System.out.println("CSLL is empty");
                  return;
            }
            Node temp = tail.next;
            while(temp!=tail) {
                  System.out.print(temp.value+" => ");
                  temp=temp.next;
            }
            System.out.println(temp.value);
      }
      void addHead(int value) {
            Node temp = new Node(value,null);
            if(size==0) {
                  tail = temp;
                  temp.next = temp;
            }
            else {
                  temp.next = tail.next;
                  tail.next = temp;
            }
            size++;
      }
      void addTail(int value) {
        Node temp = new Node(value,null);
        if(size==0) {
            tail = temp;
            temp.next = temp;
        }
        else {
            temp.next = tail.next;
            tail.next = temp;
            tail = temp;
        }
        size++;
    }
    void addPos(int pos,int value) {
        Node newNode = new Node(value,null);
        if(size==0) {
            tail = newNode;
            newNode.next = newNode;
        }
        else {
            if(pos==0) {
                Node temp = tail.next;
                newNode.next = temp;
                tail.next = newNode;
                return;
            }
            Node temp = tail.next;
            int i=0;
            while(temp.next!=tail && i<pos-1) {
                temp=temp.next;
                i++;
            }
            newNode.next = temp.next;
            temp.next = newNode;
        }
        size++;
    }
    void removeHead() {
        if(size==0) {
            System.out.println("CSLL is empty");
            return;
        }
        if(tail==tail.next)
            tail = null;
        else
            tail.next = tail.next.next;
        size--;
    }
    void removeTail() {
        if(size==0) {
            System.out.println("CSLL is empty");
            return;
        }
        if(tail==tail.next)
            tail = null;
```

```
		else
		{
			Node temp = tail.next;
			while(temp.next!=tail) {
				temp = temp.next;
			}
			temp.next = tail.next;
			tail = temp;
		}
		size--;
	}
	void deleteElement(int value) {
		if(size==0) {
			System.out.println("CSLL is empty");
			return;
		}
		Node prev=tail,currNode=tail.next,head=tail.next;
		if(currNode.value==value) {
			if(currNode==currNode.next)
				tail = null;
			else
				tail.next = tail.next.next;
			return;
		}
		prev = currNode;
		currNode = currNode.next;
		while(currNode!=head) {
			if(currNode.value == value) {
				if(currNode==tail)
					tail = prev;
				prev.next = currNode.next;
				return;
			}
			prev = currNode;
			currNode = currNode.next;
		}
		return;
	}
	boolean search(int value) {
		Node temp = tail;
		for(int i=0;i<size;i++) {
			if(temp.value==value)
				return true;
			temp = temp.next;
		}
		return false;
	}
}

circular double linked list
---------------------------
public class CDLL {
	Node head = null;
	Node tail = null;
```

```
int size = 0;
class Node{
      int value;
      Node next,prev;
      Node(int value,Node next,Node prev){
            this.value = value;
            this.next = next;
            this.prev = prev;
      }
}
void print() {
      if(size==0) {
            System.out.println("CDLL is empty");
            return;
      }
      Node temp = tail.next;
      while(temp!=tail) {
            System.out.print(temp.value+" ==> ");
            temp = temp.next;
      }
      System.out.println(temp.value);
}
void addHead(int value) {
      Node newNode = new Node(value,null,null);
      if(size==0) {
            tail = head = newNode;
            newNode.next = newNode;
            newNode.prev = newNode;
      }
      else {
            newNode.next = head;
            newNode.prev = head.prev;
            head.prev = newNode;
            newNode.prev.next = newNode;
            head = newNode;
      }
      size++;
}
void addTail(int value) {
      Node newNode = new Node(value,null,null);
      if(size==0) {
            head = tail = newNode;
            newNode.next = newNode;
            newNode.prev = newNode;
      }
      else {
            newNode.next = tail.next;
            newNode.prev = tail;
            tail.next = newNode;
            newNode.next.prev = newNode;
            tail = newNode;
      }
      size++;
}
```

```
      void removeHead() {
            if(size==0) {
                  System.out.println("CDLL is empty");
                  return;
            }
            size--;
            if(size==0) {
                  head = null;
                  tail = null;
                  return;
            }
            Node temp = head.next;
            temp.prev = tail;
            tail.next = temp;
            head = temp;
      }
      void removeTail() {
            if(size==0) {
                  System.out.println("CDLL is empty");
                  return;
            }
            size--;
            if(size==0) {
                  head=null;
                  tail=null;
                  return;
            }
            Node temp = tail.prev;
            temp.next = head;
            head.prev = temp;
            tail = temp;
      }
}
```

String Data Structure in Java:

01. Introduction to strings
02. Mutablity and Immutablity objects
03. Heap and SCP memory areas
04. java.lang.String constructors and methods
05. java.lang.StringBuffer constructors and methods
06. java.lang.StringBuilder constructors and methods
07. StringBuffer vs StringBuilder
08. java.util.StringTokenizer constructors and methods
09. Regular Expression and Applications
10. Programs on String
11. Programs on StringBuffer and StringBuilder
12. Programs on StringTokenizer

01. Introduction to strings

like other programming languages, collection or group or sequence of characters is called as string. strings are enclosed within double quotes "like this". To perform basic string level operations java has provide predefined class in lang and util packages. java strings are divided into the following four types

1. java.lang.String
2. java.lang.StringBuffer
3. java.lang.StringBuilder
4. java.util.StringTokenizer

02. Mutablity and Immutablity objects

String class objects are immutable objects, if we are trying to perform any modifications on the string object, with those modifications a new string object will be created but old content can't be modified. i.e. modifications are not allowed.

Ex: String

if we are trying to perform any modifications on the existing object, if those modifications will be performed on the same object, then such type of objects are called as mutable objects. i.e. modifications are allowed.

Ex: StringBuffer and StringBuiler

Ex:

```
class Test
{
	public static void main(String[] args)
	{
		String s = new String("abc");
		s.concat("def");
		System.out.println(s);//abc
	}
}
```

Ex:

```
class Test
{
	public static void main(String[] args)
	{
		StringBuffer sb = new StringBuffer("wx");
		sb.append("yz");
		System.out.println(sb);//wxyz
	}
}
```

== is meant for content comparing for primitives
== is meant for reference or address comapring for objects
.equals() is meant for reference or address comapring for objects (java.lang.Object)

Note1:

== is meant for address of reference comparision, .equals() method is overriden in java.lang.String class for content compaision.

Ex:

```
class Test
{
      public static void main(String[] args)
      {
            String s1 = new String("abc");
            String s2 = new String("abc");
            System.out.println(s1==s2);//false
            System.out.println(s1.equals(s2));//true
      }
}
```

Note1:

== is meant for address of reference comparision, .equals() method is not overriden in java.lang.StringBuffer class for content compaision, hence it is also meant for address comparision

Ex:

```
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb1 = new StringBuffer("wxyz");
            StringBuffer sb2 = new StringBuffer("wxyz");
            System.out.println(sb1==sb2);//false
            System.out.println(sb1.equals(sb2));//false
      }
}
```

03. Heap and SCP memory areas

Ex1:

```
String s = new String("prakash");
```

```
heap -----> 1
scp ------> 1
total ----> 2
```

In this case two objects are created one is in heap and other one is in scp (string constant pool) and "s" is always pointing to heap object.

Ex2:

```
String s = "prakash";
```

```
heap -----> 0
```

```
scp ------> 1
total ----> 1
```

In this case only one object will be created in scp area.

```
Ex3:
----
String s1 = new String("abc");
String s2 = new String("abc");
String s3 = "abc";
String s4 = "abc";

heap ------> 2
scp -------> 1
total -----> 3
```

=> Object creation in SCP is always optional, first JVM will check whether the content is already existed in SCP are not, if it is existed same reference will be reused.

```
Ex4:
----
String s = new String("abc");
s.concat("def");
s=s.concat("wxyz");

heap ----> 3
scp -----> 3
total ---> 6

Ex5:
----
String s1 = new String("spring");
s1.concat("fall");
String s2 = s1.concat("winter");
s2.concat("summer");

heap ----> 4
scp -----> 4
total ---> 8

Ex:
---
class Test
{
      public static void main(String[] args)
      {
            String s1 = new String("spring");
            s1.concat("fall");
            String s2 = s1.concat("winter");
            s2.concat("summer");
            System.out.println(s1);//spring
            System.out.println(s2);//springwinter
      }
}

Ex6:
----
String s1 = new String("i love my java");
String s2 = new String("i love my java");
String s3 = "i love my java";
String s4 = "i love my java";
String s5 = "i love "+"my java";
String s6 = "i love ";
String s7 = s6 + "my java";
final String s8 = "i love ";
String s9 = s8 + "my java";

heap ----> 3
scp -----> 3
total ---> 6

Ex:
---
class Test
{
      public static void main(String[] args)
      {
            String s1 = new String("i love my java");
            String s2 = new String("i love my java");
            System.out.println(s1==s2);//false

            String s3 = "i love my java";
            String s4 = "i love my java";

            System.out.println(s1==s3);//false
            System.out.println(s2==s3);//false
            System.out.println(s3==s4);//true

            String s5 = "i love "+"my java";
            System.out.println(s3==s5);//true
            System.out.println(s4==s5);//true

            String s6 = "i love ";
            String s7 = s6 + "my java";
            System.out.println(s4==s7);//false

            final String s8 = "i love ";
            System.out.println(s6==s8);//true

            String s9 = s8 + "my java";
            System.out.println(s4==s9);//true
      }
}

Ex7:
----
String s1 = "abc";
String s2 = s1.toUpperCase();
String s3 = s1.toLowerCase();

heap ----> 1
scp -----> 1
total ---> 2

Ex:
---
class Test
{
      public static void main(String[] args)
      {
            String s1 = "abc";
            String s2 = s1.toUpperCase();
            String s3 = s1.toLowerCase();

            System.out.println(s1==s2);//false
            System.out.println(s1==s3);//true
      }
}

Ex8:
----
String s1 = "abc";
String s2 = s1.toString();

heap ----> 0
scp -----> 1
total ---> 1

Ex:
---
class Test
{
      public static void main(String[] args)
      {
            String s1 = "abc";
            String s2 = s1.toString();

            System.out.println(s1==s2);//true
      }
}

Ex9:
----
String s1 = new String("abc");
String s2 = s1.toString();

heap ---> 1
scp ----> 1
total --> 2

class Test
```

```
{
    public static void main(String[] args)
    {
        String s1 = new String("abc");
        String s2 = s1.toString();

        System.out.println(s1==s2);//true
    }
}
```

Ex10:

```
String s1 = new String("abc");
String s2 = s1.toString();
String s3 = s1.toUpperCase();
String s4 = s1.toLowerCase();
String s5 = s1.toUpperCase();
String s6 = s3.toLowerCase();
```

```
heap ----> 4
scp -----> 1
total ---> 5
```

```
class Test
{
    public static void main(String[] args)
    {
        String s1 = new String("abc");
        String s2 = s1.toString();
        System.out.println(s1==s2);//true

        String s3 = s1.toUpperCase();
        String s4 = s1.toLowerCase();
        System.out.println(s1==s4);//true

        String s5 = s1.toUpperCase();
        System.out.println(s3==s5);//false

        String s6 = s3.toLowerCase();
        System.out.println(s1==s6);//false
    }
}
```

java.lang.String constructors and methods

String()

it is used to create an empty string object.

Ex:

```
import java.util.*;
class Test
```

```
{
      public static void main(String[] args)
      {
            String s = new String();
            System.out.println(s);//
            System.out.println(s.length());//0
            System.out.println(s.isEmpty());//true
      }
}
```

String(string)
----
it is used to create a string object with given string literal value.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = new String("vishal");
            System.out.println(s);//vishal
            System.out.println(s.length());//6
            System.out.println(s.isEmpty());//false
      }
}
```

String(char[])
----
it creates a new string object with given characters.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            char[] ch = {'w','e','l','c','o','m','e'};
            String s = new String(ch);
            System.out.println(s);//welcome
      }
}
```

String(char[],int start_index,int num_of_chars)
----
it extract the characters from start_index to number of characters

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            char[] ch = {'w','e','l','c','o','m','e'};
            //            0   1   2   3   4   5   6
            String s1 = new String(ch);
            System.out.println(s1);//welcome
            String s2 = new String(ch,3,4);
            System.out.println(s2);//come
            String s3 = new String(ch,3,2);
            System.out.println(s3);//co
      }
}
```

String(byte[])
----

it creates a new string object with given byte array [ascii values].

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            byte[] b = {97, 98, 99, 65, 66, 67, 68};
            //          0   1   2   3   4   5   6
            String s = new String(b);
            System.out.println(s);//abcABCD
      }
}
```

String(StringBuffer)
----

it is used to convert our StringBuffer object into String object.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abc");
            String s = new String(sb);
            System.out.println(s);//abc
            System.out.println(sb);//abc
      }
}
```

String(StringBuilder)
----

it is used to convert our StringBuilder object into String object.

Ex:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		StringBuilder sb = new StringBuilder("abcd");
		String s = new String(sb);
		System.out.println(s);//abcd
		System.out.println(sb);//abcd
	}
}
```

int length()
----
it returns number of characters present in the given string object.

Ex:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		String s = new String("prakash babu");
		System.out.println(s);//prakash babu
		System.out.println(s.length());//12
	}
}
```

boolean isEmpty()
----
it returns true if the given string is empty else false.

Ex:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		System.out.println("abc".isEmpty());//false
		System.out.println(" ".isEmpty());//false
		System.out.println("".isEmpty());//true
	}
}
```

boolean startsWith(string)
----
it returns true if the given string starts with another string else false

Ex:

```
import java.util.*;
```

```
class Test
{
      public static void main(String[] args)
      {
            System.out.println("java is very
easy".startsWith("python"));//false
            System.out.println("java is very
easy".startsWith("java"));//true
      }
}
```

boolean endsWith(String)
----
it returns true if the given string ends with another string else false

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            System.out.println("java is very
easy".endsWith("easy"));//true
            System.out.println("java is very
easy".endsWith("difficult"));//false
      }
}
```

boolean equals(string)
----
it returns true if the given string is equal with another string, else false. consider the case while comparing.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            System.out.println("java".equals("python"));//false
            System.out.println("java".equals("abcd"));//false
            System.out.println("java".equals("JAVA"));//false
            System.out.println("java".equals("java"));//true
      }
}
```

boolean equalsIgnoreCase(string)
----
it returns true if the given string is equal with another string, else false. it ignores case while comparing.

Ex:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		System.out.println("java".equals("python"));//false
		System.out.println("java".equals("abcd"));//false
		System.out.println("java".equalsIgnoreCase("JAVA"));//true
		System.out.println("java".equals("java"));//true
	}
}
```

char[] toCharArray():

it returns char array on the given string object

Ex:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		String s = "abcd";
		char[] ch = s.toCharArray();
		System.out.println(s);//abcd
		System.out.println(Arrays.toString(ch));//['a','b','c','d']
	}
}
```

byte[] getBytes()

it returns byte array on the given string object

Ex:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		String s = "abcd";
		byte[] b = s.getBytes();
		System.out.println(s);//abcd
		System.out.println(Arrays.toString(b));//[97,98,99,100]
	}
}
```

char charAt(int index)

it returns char located at the given index, else AIOBE

Ex:

```java
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        String s = "abcd";
        System.out.println(s);//abcd
        System.out.println(s.charAt(0));//a
        System.out.println(s.charAt(1));//b
        System.out.println(s.charAt(2));//c
        System.out.println(s.charAt(3));//d
    }
}
```

int indexOf(char)
----

it returns index value of the given character

Ex:

```java
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        String s = "abcaba";
        //          012345
        System.out.println(s);//abcaba
        System.out.println(s.indexOf('a'));//0
        System.out.println(s.indexOf('b'));//1
        System.out.println(s.indexOf('d'));//-1
    }
}
```

int lastIndexOf(char)
----

it returns last index value of the given character

Ex:

```java
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        String s = "abcaba";
        //          012345
        System.out.println(s);//abcaba
        System.out.println(s.lastIndexOf('a'));//5
        System.out.println(s.lastIndexOf('b'));//4
        System.out.println(s.lastIndexOf('d'));//-1
    }
}
```

String substring(int index)

it returns complete string starting from the given index to till end

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "java is very very easy programming";
            //          0    5  8    13   18   23
            System.out.println(s);//java is very very easy programming
            System.out.println(s.substring(5));//is very very easy
programming
            System.out.println(s.substring(13));//very easy programming
            System.out.println(s.substring(23));//programming
      }
}
```

String substring(int sindex,int eindex)

it returns complete string starting from the given index to ending index-1

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "java is very very easy programming";
            //          0    5  8    13   18   23
            System.out.println(s);//java is very very easy programming
            System.out.println(s.substring(5,7));//is
            System.out.println(s.substring(8,17));//very very
            System.out.println(s.substring(18,34));//easy programming
      }
}
```

String concat(string)

it is used to concatenate with another string. we can also use + operator for this purpose.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s1 = "wel";
            String s2 = "come";
            System.out.println(s1);//wel
            System.out.println(s2);//come
            System.out.println(s1+s2);//welcome
            System.out.println(s1.concat(s2));//welcome
            System.out.println(s1);//wel
            System.out.println(s2);//come
      }
}
```

String replace(old_char, new_char)
----

it replaces old char with given new char

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = new String("abcaba");
            System.out.println(s);//abcaba
            System.out.println(s.replace('a','x'));//xbcxbx
      }
}
```

String toUpperCase()
----

it converts the given string into upper case

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            System.out.println("java".toUpperCase());//JAVA
            System.out.println("JAVA".toUpperCase());//JAVA
            System.out.println("JaVa".toUpperCase());//JAVA
            System.out.println("JavA".toUpperCase());//JAVA
      }
}
```

String toLowerCase()
----

it converts the given string into lower case

Ex:

```
import java.util.*;
```

```
class Test
{
	public static void main(String[] args)
	{
		System.out.println("java".toLowerCase());//java
		System.out.println("JAVA".toLowerCase());//java
		System.out.println("JaVa".toLowerCase());//java
		System.out.println("JavA".toLowerCase());//java
	}
}
```

String[] split(delimtier)

it divides the given string into string[] based on given delimiter

Ex1:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		String s = "java is very easy";
		String[] ss = s.split(" ");
		System.out.println(s);//java is very easy
		System.out.println(Arrays.toString(ss));//[java, is, very, easy]
	}
}
```

Ex2:

```
import java.util.*;
class Test
{
	public static void main(String[] args)
	{
		String s = "13-1-2023";
		String[] ss = s.split("-");
		System.out.println(s);//13-1-2023
		System.out.println(Arrays.toString(ss));//[13,1,2023]
	}
}
```

java.lang.StringBuffer constructors and methods

if the content is fixed, not chaning frequently then we should go for String.

if the content is not fixed and keep on changing then we should go for StringBuffer

```
String       ---> immutable
StringBuffer ---> mutable
```

StringBuffer()
----

it creates a new string buffer object with default capacity as 16.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer();
            System.out.println(sb);//
            System.out.println(sb.length());//0
            System.out.println(sb.capacity());//16
      }
}
```

StringBuffer(String)
----

it creates a new string buffer object with given string and capcaity = len(s)+16

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abc");
            System.out.println(sb);//abc
            System.out.println(sb.length());//3
            System.out.println(sb.capacity());//3+16=19
      }
}
```

StringBuffer(int capacity)
----

it creates a new string buffer object with given capacity

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer(50);
            System.out.println(sb);//
            System.out.println(sb.length());//0
            System.out.println(sb.capacity());//50
      }
}
```

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer();
            System.out.println(sb);//
            System.out.println(sb.length());//0
            System.out.println(sb.capacity());//16
            sb.append("abcdefghijklmnop");
            System.out.println(sb);//abcdefghijklmnop
            System.out.println(sb.length());//16
            System.out.println(sb.capacity());//16
            sb.append("q");
            System.out.println(sb);//abcdefghijklmnopq
            System.out.println(sb.length());//17
            System.out.println(sb.capacity());//(16+1)*2=17*2=34
            //newcapacity = (oldcapacity+1)*2
            sb.append("abcdefghijklmnopq");
            System.out.println(sb);//abcdefghijklmnopqabcdefghijklmnopq
            System.out.println(sb.length());//34
            System.out.println(sb.capacity());//34
            sb.append("wxyz");

      System.out.println(sb);//abcdefghijklmnopqabcdefghijklmnopqwxyz
            System.out.println(sb.length());//38
            System.out.println(sb.capacity());//(34+1)*2=35*2=70

      }
}
```

int length()
----

it returns number of characters present in StringBuffer.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abcd");
            System.out.println(sb);//abcd
            System.out.println(sb.length());//4
      }
}
```

int capacity()
----

it returns max number of characters it can hold.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abcd");
            System.out.println(sb);//abcd
            System.out.println(sb.length());//4
            System.out.println(sb.capacity());//4+16=20
      }
}
```

void ensureCapacity(int capacity)

it is used to increase the capacity of string buffer object

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abcd");
            System.out.println(sb);//abcd
            System.out.println(sb.length());//4
            System.out.println(sb.capacity());//4+16=20
            sb.ensureCapacity(50);
            System.out.println(sb.capacity());//50
      }
}
```

void setLength(int length)

it sets the length of string buffer object and removes extra content.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abcd");
            System.out.println(sb);//abcd
            System.out.println(sb.length());//4
            System.out.println(sb.capacity());//4+16=20
            sb.setLength(2);
            System.out.println(sb);//ab
            System.out.println(sb.length());//2
```

```
        System.out.println(sb.capacity());//20
    }
}
```

void trimToSize()
----
it finalize the given string with number of characters existed

Ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("abcd");
        System.out.println(sb);//abcd
        System.out.println(sb.length());//4
        System.out.println(sb.capacity());//4+16=20
        sb.setLength(2);
        System.out.println(sb);//ab
        System.out.println(sb.length());//2
        System.out.println(sb.capacity());//20
        sb.trimToSize();
        System.out.println(sb);//ab
        System.out.println(sb.length());//2
        System.out.println(sb.capacity());//2
    }
}
```

char charAt(int index)
----
it returns character located at the given index value.

Ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("abcd");
        System.out.println(sb);//abcd
        System.out.println(sb.charAt(0));//a
        System.out.println(sb.charAt(1));//b
        System.out.println(sb.charAt(2));//c
        System.out.println(sb.charAt(3));//d
    }
}
```

void setCharAt(int index,char ch)
----
it perform replacement operation on the given string buffer

Ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("welkome");
        System.out.println(sb);//welkome
        sb.setCharAt(3,'c');
        System.out.println(sb);//welcome
    }
}
```

void deleteCharAt(int index)
----

it removes the character located at the given index

Ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer("welkome");
        System.out.println(sb);//welkome
        sb.deleteCharAt(3);
        System.out.println(sb);//welome
    }
}
```

StringBuffer append(object)
----

it appends i.e. adds the given object at the end of string buffer

Ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        StringBuffer sb = new StringBuffer();
        System.out.println(sb);//
        sb.append("welcome ");
        System.out.println(sb);//welome
        sb.append("python ");
        System.out.println(sb);//welome python
        sb.append("programming");
        System.out.println(sb);//welome python programming
    }
}
```

StringBuffer insert(int index,object)
----
it inserts the given object at the specified location.

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer();
            System.out.println(sb);//
            sb.append("welcome ");
            System.out.println(sb);//welome
            sb.append("python ");
            System.out.println(sb);//welome python
            sb.append("programming");
            sb.insert(7," to");
            System.out.println(sb);//welome to python programming
            sb.insert(10," java and");
            System.out.println(sb);//welome to java and python
programming
      }
}
```

StringBuffer delete(int sindex,int eindex)
----
it removes the characters from s index to ending index

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer();
            System.out.println(sb);//
            sb.append("welcome ");
            System.out.println(sb);//welome
            sb.append("python ");
            System.out.println(sb);//welome python
            sb.append("programming");
            sb.insert(7," to");
            System.out.println(sb);//welome to python programming
            sb.insert(10," java and");
            System.out.println(sb);//welome to java and python
programming
            sb.delete(15,26);
            System.out.println(sb);//welome to java programming
      }
}
```

StringBuffer reverse()
----

it reverse the given string buffer's content

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            StringBuffer sb = new StringBuffer("abcdefgh");
            System.out.println(sb);//abcdefgh
            sb.reverse();
            System.out.println(sb);//hgfedcba
      }
}
```

java.lang.StringBuilder constructors and methods
----

it is exactly same as java.lang.StringBuffer, except the following differences

StringBuffer vs StringBuilder

StringBuffer

=> 1.0 version
=> synchronized
=> only one thread is allowed
=> sequential execution
=> increase application time
=> performance improved
=> thread safe
=> deprecated/outdated

StringBuilder

=> 1.5 version
=> non-synchronized
=> multiple threads are allowed
=> parallel execution
=> decreases application time
=> performance not that good
=> not thread safe
=> not deprecated/outdated

java.util.StringTokenizer constructors and methods
----

it is an utility class existed in java.util package and StringTokenizer is used to divide the given string into tokens.

"java is very easy" ------> java, is, very, easy
"19-12-2002" -------------> 19, 12, 2002

```
"09:30:45" ---------------> 09, 30, 45

StringTokenizer st = new StringTokenizer(String)
StringTokenizer st = new StringTokenizer(String,delimiter)

int countTokens() -------> returns number of tokens
boolean hasMoreTokens() -> returns true if there is a token
String nextToken() ------> returns current token and transfer the control
to next

Ex:
---
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "java is very easy";
            StringTokenizer st = new StringTokenizer(s);
            System.out.println(st.countTokens());//4
            while(st.hasMoreTokens()){
                  System.out.println(st.nextToken());
            }
      }
}
output:
-------
java
is
very
easy

Ex2:
----
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "13-1-2023";
            StringTokenizer st = new StringTokenizer(s,"-");
            System.out.println(st.countTokens());//3
            while(st.hasMoreTokens()){
                  System.out.println(st.nextToken());
            }
      }
}

output:
-------
3
13
1
2023
```

Ex3:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "09:30:34";
            StringTokenizer st = new StringTokenizer(s,":");
            System.out.println(st.countTokens());//3
            while(st.hasMoreTokens()){
                  System.out.println(st.nextToken());
            }
      }
}
```

output:

```
3
09
30
34
```

Regular Expression and Applications

a group of strings according to a particular format or pattern or template is called as regular expression.

steps to prepare re object

1) import java.util.regex.*;

java.util -> it imports all the classes and interfaces from util pkg not sub-pkgs
java.util.regex --> it imports all the classes and interfaces from regex sub-pkg

2) pattern object (format)
3) matcher object (target)

mobile number validation:

```
      123         invalid
      1234567890  invalid
      7386237319  valid
      1386237319  invalid
```

format or re: [6-9][0-9]{9}

Ex:

```
import java.util.*;
import java.util.regex.*;
```

```
class Test
{
      public static void main(String[] args)
      {
            Pattern p = Pattern.compile("[0-9]");//re
            Matcher m = p.matcher("a1b2c9d57u");//target string
            int c=0;
            while(m.find()){
                  System.out.println(m.start()+" ====> "+m.end()+" ====>
"+m.group());
                  c++;
            }
            System.out.println("count="+c);
      }
}
```

```
1 ====> 2 ====> 1
3 ====> 4 ====> 2
5 ====> 6 ====> 9
7 ====> 8 ====> 5
8 ====> 9 ====> 7
count=5
```

Ex:

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Pattern p = Pattern.compile("[a-z]");//re
            Matcher m = p.matcher("a1b2c9d57u");//target string
            int c=0;
            while(m.find()){
                  System.out.println(m.start()+" ====> "+m.end()+" ====>
"+m.group());
                  c++;
            }
            System.out.println("count="+c);
      }
}
```

output:

```
0 ====> 1 ====> a
2 ====> 3 ====> b
4 ====> 5 ====> c
6 ====> 7 ====> d
9 ====> 10 ====> u
count=5
```

predefined character classes:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
\\s   space characters
\\S   except space character
\\d   digits
\\D   except digits
\\w   word character i.e. a-z or A-Z or 0-9
\\W   except word character (special characters)
.     all characters

Ex:
----
Pattern p = Pattern.compile("\\s");
Pattern p = Pattern.compile("\\S");
Pattern p = Pattern.compile("\\d");
Pattern p = Pattern.compile("\\D");
Pattern p = Pattern.compile("\\w");
Pattern p = Pattern.compile("\\W");
Pattern p = Pattern.compile(".");

Ex:
---
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Pattern p = Pattern.compile("\\W");//re
            Matcher m = p.matcher("ab c$123#iJk^45 6*pQr @ wXYz");
            int c=0;
            while(m.find()){
            System.out.println(m.start()+" ====> "+m.end()+" ====>
"+m.group());
            c++;
            }
            System.out.println("count="+c);
      }
}

userdefined character classes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[abc]       either 'a' or 'b' or 'c'
[^abc]            except either 'a' or 'b' or 'c'
[0-9]       all digits from 0 to 9
[^0-9]            except digits from 0 to 9
[a-z]       lower case characters
[^a-z]            except lower case characters
[A-Z]       upper case characters
[^A-Z]            except upper case characters
[a-zA-Z]    all lower case and upper case characters
[^a-zA-Z]   except lower case and upper case characters
[a-zA-Z0-9]all lower case, upper case and digits
[^a-zA-Z0-9]      except lower case, upper case and digits
```

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Pattern p = Pattern.compile("[^a-zA-Z0-9 ]");//re
            Matcher m = p.matcher("ab c$123#iJk^45 6*pQr @ wXYz");
            int c=0;
            while(m.find()){
            System.out.println(m.start()+" ====> "+m.end()+" ====> 
"+m.group());
            c++;
            }
            System.out.println("count="+c);
      }
}
```

quantifiers:
~~~~~~~~~~~~

a	exactly one a
a*	zero or more number of a's
a+	one or more number of a's
a?	zero or one a
a{m}	exactly 'm' number of a's
a{m,n}	min 'm' number of a's and max 'n' number of a's

Ex:

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Pattern p = Pattern.compile("a{3,5}");//re
            Matcher m = p.matcher("abaabaaabaaaabaaaaabaaaaaab");
            int c=0;
            while(m.find()){
            System.out.println(m.start()+" ====> "+m.end()+" ====> 
"+m.group());
            c++;
            }
            System.out.println("count="+c);
      }
}
```

25. Impl prg to validate ATM pin number.

```
import java.util.*;
import java.util.regex.*;
class Test
{
```
~~~~~~~~~~~~

```
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s.matches("[0-9]{4}"));
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
1234
true

C:\3pm>java Test
123
false

C:\3pm>java Test
12345
false

C:\3pm>java Test
a1b2
false
```

26. Impl prg to validate mobile number

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s.matches("[6-9][0-9]{9}"));
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
7386237319
true

C:\3pm>java Test
1386237319
false

C:\3pm>java Test
73862373190
false
```

27. Impl prg to validate gmail id

```
[a-z][a-z0-9_][a-z0-9_]+@gmail[.]com

import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
      System.out.println(s.matches("[a-z][a-z0-9_][a-z0-
9_]+@gmail[.]com"));
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
a@gmail.com
false

C:\3pm>java Test
ab@gmail.com
false

C:\3pm>java Test
abc@gmail.com
true

C:\3pm>java Test
abc123@gmail.com
true

C:\3pm>java Test
123abc@gmail.com
false

C:\3pm>java Test
Abc@gmail.com
false

C:\3pm>java Test
abcd123@yahoo.com
false
```

28. Impl prg to validate student university hall ticket number

```
DSxxxx ===> DS[0-9]{4}

import java.util.*;
import java.util.regex.*;
class Test
```

```
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s.matches("DS\\d{4}"));
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
DS1234
true

C:\3pm>java Test
DS12345
false

C:\3pm>java Test
TS1234
false
```

29. Impl prg to validate bike registration number

```
TS 21 AB 1234
TS[12][0-9][A-Z]{2}[0-9]{4}
```

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s.matches("TS[12][0-9][A-Z]{2}[0-9]{4}"));
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
TS22AB1234
true

C:\3pm>java Test
TS90AB1234
false
```

30. Impl prg to validate given date.

```
20-01-2023
```

```
[0123][0-9]-[01][0-9]-202[3-9]
```

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s.matches("[0123][0-9]-[01][0-9]-202[3-
9]"));
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
20-01-2023
true

C:\3pm>java Test
45-01-2023
false

C:\3pm>java Test
39-01-2023
true

C:\3pm>java Test
20-90-2023
false

C:\3pm>java Test
20-01-2022
false
```

01. Impl prg to read str and print char and corresponding index value.

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s);
            for(int i=0;i<s.length();i++){
                  System.out.println("index= "+i+" and char=
"+s.charAt(i));
            }
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
programming
programming
index= 0 and char= p
index= 1 and char= r
index= 2 and char= o
index= 3 and char= g
index= 4 and char= r
index= 5 and char= a
index= 6 and char= m
index= 7 and char= m
index= 8 and char= i
index= 9 and char= n
index= 10 and char= g
```

```
02. Impl prg to read str and print chars present at even/odd index
values.
------------------------------------------------------------------------
-
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s);
            for(int i=0;i<s.length();i++){
                  if(i%2==0)//i%2!=0
                  System.out.println("index= "+i+" and char=
"+s.charAt(i));
            }
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
python
python
index= 0 and char= p
index= 2 and char= t
index= 4 and char= o
```

```
03. Impl prg to print vowels/consonants present in the given str.
----------------------------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s);
            for(int i=0;i<s.length();i++){
                  char ch = s.charAt(i);
                  if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
                  //if(!(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'))
                        System.out.println(ch);
            }
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
prakash
prakash
a
a
```

04. Impl prg to count numbers of vowels/consonants present in the given str.

```
-------------------------------------------------------------------------
---
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            System.out.println(s);
            int c=0;
            for(int i=0;i<s.length();i++){
                  char ch = s.charAt(i);
                  if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
                        c++;
            }
            System.out.println(c);
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
welcome
welcome
3
```

05. Impl prg to sort all the characters in asc/desc order.

```
----------------------------------------------------------
```

```
import java.util.*;
import java.util.regex.*;
class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		String s = obj.nextLine();
		System.out.println(s);
		char[] ch = s.toCharArray();
		Arrays.sort(ch);
		String ss = new String(ch);
		System.out.println(ss);
	}
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
welcome
welcome
ceelmow
```

06. Impl prg to check whether the given strs are anagrams or not.

"race" and "care"

4 and 4

['a','c','e','r'] and ['a','c','e','r']

```
import java.util.*;
import java.util.regex.*;
class Test
{
	public static void main(String[] args)
	{
		String s1 = "race";
		String s2 = "care";
		String s3 = "cary";

		char[] ch1 = s1.toCharArray();
		char[] ch2 = s2.toCharArray();
		char[] ch3 = s3.toCharArray();

		Arrays.sort(ch1);
		Arrays.sort(ch2);
		Arrays.sort(ch3);

		System.out.println(Arrays.equals(ch1,ch2));//true
		System.out.println(Arrays.equals(ch1,ch3));//false
	}
}
```

07. Impl prg to check whether the given str is paliandrome or not.

```
------------------------------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "madam";
            String ss = new StringBuffer(s).reverse().toString();

            System.out.println(s.equals(ss));//true
      }
}
```

08. Impl prg to check whether the given str is pangram or not.

```
--------------------------------------------------------------
all english alphabets should be there in that string

s = "abcdefghijklmnopqrstuvwxyz"        true
s = "abcdefghijkmnopqrstuvwxyz"               false
s = "the quick brown fox jumps over lazy dog" true

import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            boolean flag = true;
            for(int i='a';i<='z';i++){
                  if(s.indexOf(i)<0)
                  {
                        flag=false;
                        break;
                  }
            }
            System.out.println(flag);//true
      }
}
```

09. Impl prg to divide the strings seperated by spaces/comma/-.

```
---------------------------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringTokenizer st = new StringTokenizer(s);
            while(st.hasMoreTokens())
                  System.out.println(st.nextToken());
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
the
quick
brown
fox
jumps
over
lazy
dog
```

```
10. Impl prg to reverse the entire sentence.
--------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringBuffer sb = new StringBuffer(s);
            sb.reverse();
            System.out.println(sb);
      }
}

C:\3pm>javac Test.java

C:\3pm>java Test
god yzal revo spmuj xof nworb kciuq eht
```

```
11. Impl prg to reverse individual words.
-----------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringTokenizer st = new StringTokenizer(s);
            while(st.hasMoreTokens())
            System.out.print(new StringBuffer(st.nextToken()).reverse()+"
");
      }
}

C:\3pm>java Test
eht kciuq nworb xof spmuj revo yzal god
C:\3pm>
```

12. Impl prg to reverse alternative words.

```
------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringTokenizer st = new StringTokenizer(s);
            int i=0;
            System.out.println(s);
            while(st.hasMoreTokens())
            {
            if(i%2==0)
            System.out.print(st.nextToken()+" ");
            else
            System.out.print(new StringBuffer(st.nextToken()).reverse()+"
");
            i++;
            }
      }
}

C:\3pm>java Test
the quick brown fox jumps over lazy dog
the kciuq brown xof jumps revo lazy god
```

13. Impl prg to reverse even/odd length words.

```
----------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringTokenizer st = new StringTokenizer(s);
            StringBuffer sb = new StringBuffer();
            int i=0;
            System.out.println(s);
            while(st.hasMoreTokens())
            {
                  String ss = st.nextToken();
                  if(ss.length()%2==0)
                        sb.append(new StringBuffer(ss).reverse());
                  else
                        sb.append(ss);
                  sb.append(" ");
            }
            System.out.println(sb);
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
the quick brown fox jumps over lazy dog
the quick brown fox jumps revo yzal dog

14. Impl prg to convert every word first char into caps.
--------------------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringTokenizer st = new StringTokenizer(s);
            StringBuffer sb = new StringBuffer();
            System.out.println(s);
            while(st.hasMoreTokens())
            {
                  String ss = st.nextToken();

      sb.append(ss.substring(0,1).toUpperCase()+ss.substring(1));
                  sb.append(" ");
            }
            System.out.println(sb);
      }
}

C:\3pm>java Test
the quick brown fox jumps over lazy dog
The Quick Brown Fox Jumps Over Lazy Dog

15. Impl prg to convert every word first and last char into caps.
-----------------------------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            String s = "the quick brown fox jumps over lazy dog";
            StringTokenizer st = new StringTokenizer(s);
            StringBuffer sb = new StringBuffer();
            System.out.println(s);
            while(st.hasMoreTokens())
            {
                  String ss = st.nextToken();
                  int n=ss.length();

      sb.append(ss.substring(0,1).toUpperCase()+ss.substring(1,n-
1)+ss.substring(n-1,n).toUpperCase());
                  sb.append(" ");
```

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


C:\3pm>javac Test.java

C:\3pm>java Test
the quick brown fox jumps over lazy dog
ThE QuicK BrowN FoX JumpS OveR LazY DoG
```

16. Impl prg to convert except first and last chars, remaining into upper case.

```
import java.util.*;
import java.util.regex.*;
class Test
{
    public static void main(String[] args)
    {
        String s = "the quick brown fox jumps over lazy dog";
        StringTokenizer st = new StringTokenizer(s);
        StringBuffer sb = new StringBuffer();
        System.out.println(s);
        while(st.hasMoreTokens())
        {
            String ss = st.nextToken();
            int n=ss.length();
            sb.append(ss.substring(0,1)+ss.substring(1,n-
1).toUpperCase()+ss.substring(n-1,n));
            sb.append(" ");
        }
        System.out.println(sb);
    }
}
```

output:

```
C:\3pm>javac Test.java

C:\3pm>java Test
the quick brown fox jumps over lazy dog
tHe qUICk bROWn fOx jUMPs oVEr lAZy dOg
```

17. American keyboard

Given a string, return the true if that can be typed using letters of alphabet on only one row's of American keyboard like the image below.

In the American keyboard:
=> the first row consists of the characters "qwertyuiop",
=> the second row consists of the characters "asdfghjkl", and

=> the third row consists of the characters "zxcvbnm".

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();

            String r1 = "qwertyuiop";
            String r2 = "asdfghjkl";
            String r3 = "zxcvbnm";

            int c1=0,c2=0,c3=0;

            for(int i=0;i<s.length();i++){
                  if(r1.contains(s.charAt(i)+""))
                        c1++;
                  if(r2.contains(s.charAt(i)+""))
                        c2++;
                  if(r3.contains(s.charAt(i)+""))
                        c3++;
            }


      System.out.println(c1==s.length()||c2==s.length()||c3==s.length());
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
mom
false

C:\3pm>java Test
dad
true

C:\3pm>java Test
false
false

C:\3pm>java Test
true
true
```

18. Rotate String

Given two strings s and ss, return true if and only if s can become ss after some number of shifts on s. A shift on s consists of moving the leftmost character of s to the rightmost position.
For example, if s = "abcde", then it will be "bcdea" after one shift.

```
s = "abcde"

"abcde"
"bcdea"
"cdeab"
"deabc"
"eabcd"

"bcdea" ---> true
"bdcea" ---> false

"abcdeabcde".contains(ss)
```

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            String ss = obj.nextLine();
            System.out.println((s+s).contains(ss));
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
abcde
bcdea
true

C:\3pm>java Test
abcde
bdcea
false
```

19. Impl prg to return middle char(s).

```
--------------------------------------
abc ----> b
abcd ---> bc
0123
4 ---> n/2-1 and n/2
3 ---> n/2
```

```
import java.util.*;
import java.util.regex.*;
```

```
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            int n=s.length();
            if(n%2==0)
                  System.out.println(s.charAt(n/2-1)+""+s.charAt(n/2));
            else
                  System.out.println(s.charAt(n/2));
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
abc
b

C:\3pm>java Test
abcd
bc
```

20. Impl prg to remove duplicate characters from the given str.

"welcome" ----> "welcom"
"abcaba" -----> "abc"

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            String ss = "";
            for(int i=0;i<s.length();i++)
            {
                  if(ss.indexOf(s.charAt(i))<0)
                        ss=ss+s.charAt(i);
            }
            System.out.println(ss);
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
welcome
welcom

C:\3pm>java Test
```

```
abcaba
abc

C:\3pm>java Test
prakash
praksh
```

21. Chess Board

You are given coordinates, a string that represents the coordinates of a square of the chess board. bellow is the chess board for your reference.

Return True if the saquare is in white, and false if the square is in Black.

The coordinates will always represent a valid chess board square. The coordinates will always have the letter first, and the number second.

```
import java.util.*;
import java.util.regex.*;
class Test
{
    public static void main(String[] args)
    {
        Scanner obj = new Scanner(System.in);
        String s = obj.nextLine();
        int x = s.charAt(0)-96;
        int y = s.charAt(1);
        System.out.println((x+y)%2!=0);
    }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
a1
false

C:\3pm>java Test
a2
true

C:\3pm>java Test
f7
true

C:\3pm>java Test
h4
false
```

22. Impl prg to convert lower case chars to upper case and vice versa (swapcase).

```
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            String ss="";
            for(int i=0;i<s.length();i++)
            {
                  char ch = s.charAt(i);
                  if(ch>='a' && ch<='z')
                        ss=ss+(char)(ch-32);
                  if(ch>='A' && ch<='Z')
                        ss=ss+(char)(ch+32);
            }
            System.out.println(ss);
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
PraKasH
pRAkASh
```

23. Impl prg to remove special characters present in the given str.

```
-------------------------------------------------------------------
import java.util.*;
import java.util.regex.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            String s = obj.nextLine();
            String ss="";
            for(int i=0;i<s.length();i++)
            {
                  char ch = s.charAt(i);

      if((ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z')||(ch>='0'&&ch<='9'))
                        ss=ss+ch;
            }
            System.out.println(ss);
      }
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
pra$s^h
```

prash

24. Impl prg to convert the given integer value into english word.

```
import java.util.*;
import java.util.regex.*;
class Test
{
	public static void main(String[] args)
	{
		Scanner obj = new Scanner(System.in);
		String s = obj.nextLine();
		for(int i=0;i<s.length();i++)
		{
			char ch = s.charAt(i);
			switch(ch){
				case '0':System.out.print("zero ");break;
				case '1':System.out.print("one ");break;
				case '2':System.out.print("two ");break;
				case '3':System.out.print("three ");break;
				case '4':System.out.print("four ");break;
				case '5':System.out.print("five ");break;
				case '6':System.out.print("six ");break;
				case '7':System.out.print("seven ");break;
				case '8':System.out.print("eight ");break;
				case '9':System.out.print("nine ");break;
			}
		}
	}
}
```

```
C:\3pm>javac Test.java

C:\3pm>java Test
123
one two three
C:\3pm>java Test
56901
five six nine zero one
```

stack data structures:
~~~~~~~~~~~~~~~~~~~~~~

01. Introduction
02. Operations on stack
03. implementation of stack using arrays
04. implementation of stack using linked list
05. predefined implementation stack class (java.util.Stack)
06. toString() method implementation for stack class
07. sorted insertion into stack
08. sorting stack elements
09. bottom insertion
10. reverse of the stack
11. balanced parethesis application
12. infix, prefix and postfix expressions
~~~~~~~~~~~~~~~~~~~~~~

13. infix to postfix conversion
14. infix to prefix conversion
15. postfix evaluation

introduction

stack is a basic linear data structure that organizes elements in last-in-first-out lifo
the object which is inserted last will be the object that has to remove first.

Ex:

stack of plates

a stack allows you to access only one element from only one direction i.e. top of the stack, when we are inserting a new object into the stack top will be incremented by one unit and when we are removing an object from stack top will be decremented by one unit.

common operations on stack

The following are the common operations that can be performed on stack.

1. push(), inserting an element into stack
2. pop(), removing an element from stack
3. peek(), returning top most element from stack
4. isEmpty(), return true if the stack is empty
5. size(), return size of the stack i.e. number of elements
6. search(), return true if the element is existed in the stack

the following are the various representation we can use for stacks

1. implementation of stack using arrays
2. implementation of stack using linked list
3. implementation of stack using predefined library

implementation of stack using arrays

```
import java.util.*;

class StackArray{
      int size = 5;
      int[] data;
      int top = -1;
      StackArray(){
            data = new int[size];
      }

      boolean isEmpty(){
            return top==-1;
      }

      int getSize(){
            return top+1;
```

```
}

void print(){
      if(isEmpty()){
            System.out.println("stack under flow");
            return;
      }
      else{
            for(int i=0;i<=top;i++)
                  System.out.print(data[i]+" ");
            System.out.println();
      }
}

void push(int value){
      if(getSize()==data.length){
            System.out.println("stack over flow");
            return;
      }
      else{
            top++;
            data[top] = value;
      }
}

int pop(){
      if(isEmpty()){
            System.out.println("stack is under flow");
            return -1;
      }
      else{
            int value = data[top];
            top--;
            return value;
      }
}

int peek(){
      if(isEmpty()){
            System.out.println("stack is under flow");
            return -1;
      }
      else{
            return data[top];
      }
}

boolean search(int value){
      if(isEmpty()){
            System.out.println("under flow ");
            return false;
      }
      else{
            for(int i=0;i<=top;i++){
                if(data[i]==value)
                    return true;
            }
            return false;
        }
    }
}

class Test
{
    public static void main(String[] args)
    {
        Scanner obj = new Scanner(System.in);
        StackArray s = new StackArray();
        s.push(111);
        s.push(222);
        s.push(333);
        s.push(444);
        s.push(555);
        s.print();
        System.out.println(s.pop());
        s.print();
        System.out.println(s.peek());
        System.out.println(s.search(333));
        System.out.println(s.search(999));
    }
}
```

```
output:
-------
111 222 333 444 555
555
111 222 333 444
444
true
false
```

```
implementation of stack using linked list
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;

class StackLL
{
    Node head=null;
    int size = 0;
    class Node{
        int value;
        Node next;
        Node(int value,Node next){
            this.value = value;
            this.next = next;
        }
    }
    int getSize(){
```

```
            return this.size;
      }
      boolean isEmpty(){
            return size==0;
      }
      void print(){
            Node temp=head;
            if(isEmpty()){
                  System.out.println("stack is empty");
                  return;
            }
            while(temp!=null){
                  System.out.print(temp.value+" ");
                  temp = temp.next;
            }
            System.out.println();
      }
      void push(int value){
            head = new Node(value,head);
            size++;
      }
      int peek(){
            if(isEmpty())
                  return -1;
            else
                  return head.value;
      }
      int pop(){
            if(isEmpty()){
                  System.out.println("stack is under flow");
                  return -1;
            }
            else{
                  int temp = head.value;
                  head = head.next;
                  return temp;
            }
      }
}

class Test
{
      public static void main(String[] args)
      {
            StackLL s = new StackLL();
            s.push(111);
            s.push(222);
            s.push(333);
            s.push(444);
            s.print();
            System.out.println(s.peek());//
            System.out.println(s.pop());//444
            System.out.println(s.pop());//333
            s.print();
```

```
	}
}

predefined implementation stack class (java.util.Stack)
-------------------------------------------------------
case1: general implementation
-----------------------------
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Stack s = new Stack();
		System.out.println(s.empty());//true
		s.push(111);
		s.push(222);
		s.push(333);
		s.push(444);
		s.push(555);
		System.out.println(s);//[111,222,333,444,555]
		System.out.println(s.peek());//555
		System.out.println(s.pop());//555
		System.out.println(s);//[111,222,333,444]
		System.out.println(s.search(333));//2
		System.out.println(s.search(555));//-1
	}
}

case2: to hold String objects
-----------------------------
import java.util.*;

class Test
{
	public static void main(String[] args)
	{
		Stack<String> s = new Stack<String>();
		System.out.println(s.empty());//true
		s.push("AAA");
		s.push("BBB");
		s.push("CCC");
		s.push("DDD");
		System.out.println(s.empty());//false
		System.out.println(s.peek());//DDD
		System.out.println(s.search("CCC"));//2
		System.out.println(s.pop());//DDD
		System.out.println(s);//[AAA,BBB,CCC]
	}
}

case3: to hold student class objects
------------------------------------
import java.util.*;
```

```
class Student{
      int sid;
      String name;
      Student(int sid,String name){
            this.sid = sid;
            this.name = name;
      }
      public String toString(){
            return "("+sid+","+name+")";
      }
}

class Test
{
      public static void main(String[] args)
      {
            Stack<Student> s = new Stack<Student>();

            Student s1 = new Student(444,"BBB");
            Student s2 = new Student(111,"AAA");
            Student s3 = new Student(555,"EEE");
            Student s4 = new Student(333,"DDD");
            Student s5 = new Student(222,"XXX");

            s.push(s1);
            s.push(s2);
            s.push(s3);
            s.push(s4);
            s.push(s5);

            System.out.println(s);

      }
}
```

```
C:\prakashclasses>java Test
[(444,BBB), (111,AAA), (555,EEE), (333,DDD), (222,XXX)]
```

toString() method implementation for stack class

```
import java.util.*;

class StackLL
{
      Node head=null;
      int size = 0;
      class Node{
            int value;
            Node next;
            Node(int value,Node next){
                  this.value = value;
                  this.next = next;
            }
      }
      int getSize(){
            return this.size;
      }
      boolean isEmpty(){
            return size==0;
      }
      void print(){
            Node temp=head;
            if(isEmpty()){
                  System.out.println("stack is empty");
                  return;
            }
            while(temp!=null){
                  System.out.print(temp.value+" ");
                  temp = temp.next;
            }
            System.out.println();
      }
      void push(int value){
            head = new Node(value,head);
            size++;
      }
      int peek(){
            if(isEmpty())
                  return -1;
            else
                  return head.value;
      }
      int pop(){
            if(isEmpty()){
                  System.out.println("stack is under flow");
                  return -1;
            }
            else{
                  int temp = head.value;
                  head = head.next;
                  return temp;
            }
      }
      public String toString(){
            Node temp = head;
            StringBuffer sb = new StringBuffer();
            sb.append("[");
            while(temp!=null){
                  if(temp.next!=null)
                        sb.append(temp.value+", ");
                  else
                        sb.append(temp.value);
                  temp = temp.next;
            }
            sb.append("]");
            return sb.toString();
```

```
    }
}

class Test
{
    public static void main(String[] args)
    {
        StackLL s = new StackLL();
        s.push(111);
        s.push(222);
        s.push(333);
        s.push(444);
        s.print();
        System.out.println(s);
    }
}
```

```
C:\prakashclasses>java Test
444 333 222 111
[444, 333, 222, 111]
```

sorted insertion into stack

```
import java.util.*;

class Test
{
    static void sortedInsert(Stack<Integer> ss,int value){
        int temp;
        if(ss.empty() || value>ss.peek())
            ss.push(value);
        else{
            temp = ss.pop();
            sortedInsert(ss,value);
            ss.push(temp);
        }
    }
    public static void main(String[] args)
    {
        Stack<Integer> s = new Stack<Integer>();
        s.push(1);
        s.push(3);
        s.push(4);
        System.out.println(s);//[1,3,4]
        sortedInsert(s,2);
        System.out.println(s);//[1,2,3,4]
    }
}
```

sorting stack elements

```
import java.util.*;
```

```
class Test
{

      static void sortedInsert(Stack<Integer> ss,int value){
            int temp;
            if(ss.empty() || value>ss.peek())
                  ss.push(value);
            else{
                  temp = ss.pop();
                  sortedInsert(ss,value);
                  ss.push(temp);
            }
      }
      static void sortStack(Stack<Integer> ss){
            int temp;
            if(ss.empty()==false){
                  temp = ss.pop();
                  sortStack(ss);
                  sortedInsert(ss,temp);
            }
      }
      public static void main(String[] args)
      {
            Stack<Integer> s = new Stack<Integer>();
            s.push(5);
            s.push(3);
            s.push(1);
            s.push(4);
            s.push(2);
            System.out.println(s);//[5,3,1,4,2]
            sortStack(s);
            System.out.println(s);//[1,2,3,4,5]
      }
}

bottom insertion
----------------
import java.util.*;

class Test
{
      static void bottomInsert(Stack<Integer> ss, int value){
            int temp;
            if(ss.empty())
                  ss.push(value);
            else{
                  temp = ss.pop();
                  bottomInsert(ss,value);
                  ss.push(temp);
            }
      }

      public static void main(String[] args)
      {
            Stack<Integer> s = new Stack<Integer>();
            s.push(1);
            s.push(2);
            s.push(3);
            System.out.println(s);//[1,2,3]
            bottomInsert(s,999);
            bottomInsert(s,888);
            System.out.println(s);//[888,999,1,2,3]
      }
}
```

reverse of the stack

```
import java.util.*;

class Test
{
    static void bottomInsert(Stack<Integer> ss, int value){
        int temp;
        if(ss.empty())
            ss.push(value);
        else{
            temp = ss.pop();
            bottomInsert(ss,value);
            ss.push(temp);
        }
    }
    static void reverseStack(Stack<Integer> ss){
        if(ss.empty())
            return;
        else{
            int temp = ss.pop();
            reverseStack(ss);
            bottomInsert(ss,temp);
        }
    }
    public static void main(String[] args)
    {
        Stack<Integer> s = new Stack<Integer>();
        s.push(1);
        s.push(2);
        s.push(3);
        System.out.println(s);//[1,2,3]
        reverseStack(s);
        System.out.println(s);//[3,2,1]
    }
}
```

balanced parethesis application

```
() yes
[] yes
(()) yes
([}) no
```

```java
import java.util.*;

class Test
{
      static boolean isBalancedParenthesis(String exp){
            Stack<Character> s = new Stack<Character>();
            for(char ch:exp.toCharArray()){
                  switch(ch){
                  case '(':
                  case '[':
                  case '{':
                                    s.push(ch); break;
                  case ')':
                                    if(s.pop()!='(')
                                          return false;
                                    break;
                  case ']':
                                    if(s.pop()!='[')
                                          return false;
                                    break;
                  case '}':
                                    if(s.pop()!='{')
                                          return false;
                                    break;
                  }
            }
                  return s.empty();
      }

      public static void main(String[] args)
      {
            System.out.println(isBalancedParenthesis("()"));//true
            System.out.println(isBalancedParenthesis("()[{}"));//false
      }
}
```

12. infix, prefix and postfix expressions

Expression is a combination of operators and operands. all the expressions in programming are divided into the following three types.

1) infix expression =======> operand OPERATOR operand
2) prefix expression ======> OPERATOR operand operand
3) postfix expression =====> operand operand OPERATOR

Ex: expr ---> a+b

```
      infix ----> a+b
      postfix---> ab+
      prefix ---> +ab
```

13. infix to postfix conversion

infix to postfix conversion examples

Ex1: a+b

```
input -----> a, +,
stack -----> +
output ----> ab+
```

postfix: ab+

Ex2: a*b+c

```
input -----> a, *, b, +, c
stack -----> *,+
output ----> ab*c+
```

postfix: ab*c+

Ex3: a+b*c

```
input ------> a, +, b, *, c
stack ------> +, *
output -----> abc*+
```

postfix: abc*+

Ex4: a+b-c

```
input-----> a, +, b, -, c
stack-----> +, -
output ---> abc-+
```

postfix: abc-+

Ex5: a+b/c-d

```
input -----> a, +, b, /, c,d
stack -----> +,/
output ----> abc/d-+
```

postfix: abc/d-+

Ex6: (a+b)*(c+d)

```
input -----> (, a, +, b, ), *, (, c, +, d
stack -----> *(+
output ----> ab+cd+*
```

postfix: ab+cd+*

infix to postfix conversion algorithm

1) read the expression from left to right.
2) if the input symbol is '(' then push it into the stack.

3) if the input symbol is an operand then put it into output.

4) if the input symbol is an operator then,

i) check the precedence of an operator which is existed inside the stack is having greater precedence then the precedence of incoming symbol, then remove that operator from stack and push it into output. repeat this process till you get the operator which is having less priority.

ii) otherwise push that operator into stack.

5) if the input symbol is ')' then pop all operators from stack, place them in the output till the openining parenthsis is encountered. dn't push parenthsis into the output.

6) if all the symbols are extracted then pop all items from stack and push it into output.

7) print the output.

implementation of infix to postfix algorithm

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Test
{
      static int precedence(char ch){
            if(ch=='*' || ch=='/')
                  return 2;
            if(ch=='+' || ch=='-')
                  return 1;
            return -1;
      }
      static String infixToPostfix(String s){
            String output = "";
            Stack<Character> stack = new Stack<Character>();
            for(int i=0;i<s.length();i++){
                  char ch = s.charAt(i);
                  if(Character.isLetterOrDigit(ch))
                        output = output + ch;
                  else if(ch=='(')
                        stack.push(ch);
                  else if(ch==')'){
                        while(!stack.empty() && stack.peek()!='('){
                              output = output + ch;
                              stack.pop();
                        }
                        stack.pop();
                  }
                  else{
                        while(!stack.empty() &&
precedence(ch)<=precedence(stack.peek())){
                              output = output + stack.peek();
                              stack.pop();
                        }
                        stack.push(ch);
                  }
            }
            while(!stack.empty()){
```

```
                  output = output + stack.peek();
                  stack.pop();
            }
            return output;
      }
      public static void main(String[] args)
      {
            System.out.println(infixToPostfix("a+b"));//ab+
            System.out.println(infixToPostfix("a*b+c"));//ab*c+
            System.out.println(infixToPostfix("a+b*c"));//abc*+
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
ab+
ab*c+
abc*+
```

14. infix to prefix conversion

infix to prefix conversion example

Ex:

```
      expr:  a+b

      1) reverse given expr -----> b+a
      2) postfix conv -----------> ba+
      3) reverse postfix --------> +ab

      prefix: +ab
```

infix to prefix conversion algorithm

1) reverse the given input expression.
2) apply infix to postfix conversion method.
3) reverse the output expression.

infix to prefix conversion implementation

```
import java.util.*;
class Test
{
      static int precedence(char ch){
            if(ch=='*' || ch=='/')
                  return 2;
            if(ch=='+' || ch=='-')
                  return 1;
            return -1;
      }
      static String infixToPostfix(String s){
            String output = "";
            Stack<Character> stack = new Stack<Character>();
            for(int i=0;i<s.length();i++){
                  char ch = s.charAt(i);
                  if(Character.isLetterOrDigit(ch))
                        output = output + ch;
                  else if(ch=='(')
                        stack.push(ch);
                  else if(ch==')'){
                        while(!stack.empty() && stack.peek()!='('){
                              output = output + ch;
                              stack.pop();
                        }
                        stack.pop();
                  }
                  else{
                        while(!stack.empty() &&
precedence(ch)<=precedence(stack.peek())){
                              output = output + stack.peek();
                              stack.pop();
                        }
                        stack.push(ch);
                  }
            }
            while(!stack.empty()){
                  output = output + stack.peek();
                  stack.pop();
            }
            return output;
      }
      public static String infixToPrefix(String s){
            String output = new String();
            output = new StringBuffer(s).reverse().toString();
            output = infixToPostfix(output);
            output = new StringBuffer(output).reverse().toString();
            return output;
      }
      public static void main(String[] args)
      {
            System.out.println(infixToPostfix("a+b"));//ab+
            System.out.println(infixToPostfix("a*b+c"));//ab*c+
            System.out.println(infixToPostfix("a+b*c"));//abc*+

            System.out.println(infixToPrefix("a+b"));//+ab
            System.out.println(infixToPrefix("a*b+c"));//+*abc
            System.out.println(infixToPrefix("a+b*c"));//+a*bc
      }
}
```

15. postfix evaluation

The postfix notation is used to represent algebric expr. The expressions written in postfix form are evaluated faster compared with normal infix notation.

```
steps:
------
1) create a stack to store operands
2) scan the expr from left to right and do the following operations
   i) if the element is a number, push it into the stack.
   ii) if the element is an operator, pop two operands from the stack,
evaluate that operation and push the result back to the stack.
3) when an expr is ended, the number in stack is the result.

Ex:
---
      exp: a+b ---> 2+3

      postfix: ab+ ---> 23+

      result: 5


implementation:
---------------
import java.util.*;
class Test
{
      static int precedence(char ch){
            if(ch=='*' || ch=='/')
                  return 2;
            if(ch=='+' || ch=='-')
                  return 1;
            return -1;
      }
      static String infixToPostfix(String s){
            String output = "";
            Stack<Character> stack = new Stack<Character>();
            for(int i=0;i<s.length();i++){
                  char ch = s.charAt(i);
                  if(Character.isLetterOrDigit(ch))
                        output = output + ch;
                  else if(ch=='(')
                        stack.push(ch);
                  else if(ch==')'){
                        while(!stack.empty() && stack.peek()!='('){
                              output = output + ch;
                              stack.pop();
                        }
                        stack.pop();
                  }
                  else{
                        while(!stack.empty() &&
precedence(ch)<=precedence(stack.peek())){
                              output = output + stack.peek();
                              stack.pop();
                        }
                        stack.push(ch);
```

```
                    }
                }
                while(!stack.empty()){
                    output = output + stack.peek();
                    stack.pop();
                }
                return output;
        }
        public static int evalPostfix(String s){
                Stack<Integer> stack = new Stack<Integer>();
                for(int i=0;i<s.length();i++){
                    char ch = s.charAt(i);
                    if(Character.isDigit(ch))
                        stack.push(ch-'0');
                    else {
                        int v1 = stack.pop();
                        int v2 = stack.pop();
                        switch(ch){
                        case '+': stack.push(v2+v1); break;
                        case '-': stack.push(v2-v1); break;
                        case '*': stack.push(v2*v1); break;
                        case '/': stack.push(v2/v1); break;
                        }
                    }
                }
                return stack.pop();
        }
        public static void main(String[] args)
        {
                System.out.println(evalPostfix(infixToPostfix("2+3")));
        }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
5
```

Queue data structure:

~~~~~~~~~~~~~~~~~~~~

Queue is a linear data structure which follows FIFO method, First-In-First-Out method. the item which we are inserting first will be the item which we are removing first. We will maintain two seperate pointer for insert and delete operations front and rear. when an item is inserted into the queue then rear pointer will be increamented by one unit and when an item is removed from the queue then front pointer will be increamented by one unit.

The following are the various operations that we can able to perform on queue

1) inserting an object into queue
2) deleting an object from queue
~~~~~~~~~~~~~~~~~~~~

3) displaying all the objects in a queue

These queues are divided into three types

1) normal queue
2) circular queue
3) dequeue

normal queue implementation by using arrays

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;

class NQ{
      int front,rear,size,Q[];
      NQ(){
            front = -1;
            rear = -1;
            size = 5;
            Q = new int[size];
      }
      void insert(int value){
            if(rear==size){
                  System.out.println("NQ is full");
                  return;
            }
            if(front==rear)
                  front=rear=0;
            Q[rear++]=value;
      }
      void delete(){
            if(front==rear){
                  System.out.println("NQ is empty");
                  return;
            }
            System.out.println("Deleted object is: "+Q[front]);
            front++;
            if(front==rear)
                  front=rear=-1;
      }
      void display(){
            if(front==rear){
                  System.out.println("NQ is empty");
                  return;
            }
            for(int i=front;i<rear;i++){
                  System.out.print(Q[i]+" ");
            }
            System.out.println();
      }
}

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

```
      {
            NQ q = new NQ();
            q.insert(111);
            q.insert(222);
            q.insert(333);
            q.insert(444);
            q.insert(555);
            q.insert(666);
            q.display();
            q.delete();
            q.display();
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
NQ is full
111 222 333 444 555
Deleted object is: 111
222 333 444 555
```

```
normal queue implementation by using linked list
------------------------------------------------
import java.util.Scanner;
class NQList
{
      Node front,rear;
      int size;
      class Node
      {
            int data;
            Node next;
            Node(int data,Node next){
                  this.data = data;
                  this.next = next;
            }
      }
      NQList(){
            front = null;
            rear = null;
            size = 0;
      }
      void display(){
            if(size==0){
                  System.out.println("q is empty");
                  return;
            }
            Node temp = front;
            while(temp!=null){
                  System.out.print(temp.data+" ");
                  temp = temp.next;
            }
            System.out.println();
      }
      void insert(int value){
            Node newNode = new Node(value,null);
            if(front==null && rear==null){
                  front = newNode;
                  rear = newNode;
            }
            else{
                  rear.next = newNode;
                  rear = newNode;
            }
            size++;
      }
      void delete(){
            if(size==0){
                  System.out.println("q is empty");
                  return;
            }
            System.out.println("Deleted item is: "+front.data);
            front = front.next;
            size--;
      }
}
class Test
{
      public static void main(String[] args)
      {
            NQList q = new NQList();
            q.display();//q is empty
            q.insert(111);
            q.insert(222);
            q.insert(333);
            q.display();//111, 222, 333
            q.delete();//111
            q.display();//222, 333
            q.delete();//222
            q.display();//333
            q.delete();
            q.display();//q is empty
      }
}
```

```
C:\DSAJ>javac Test.java

C:\DSAJ>java Test
q is empty
111 222 333
Deleted item is: 111
222 333
Deleted item is: 222
333
Deleted item is: 333
```

```
q is empty
```

circular queue:

There are some problems are there with normal queues, to overcomes those problems we are using circular queue. even though we have empty location inside queue, some times it will display saying "Q is full".

circular queue implementation by using arrays:

```
import java.util.*;

class CQArrays{
      int front,rear,size,c,Q[];
      CQArrays(){
            front = -1;
            rear = -1;
            c = 0;
            size = 5;
            Q = new int[size];
      }
      void insert(int value){
            if(c==size){
                  System.out.println("Q is full");
                  return;
            }
            if(front==-1)
                  front=rear=0;
            else
                  rear = (rear+1)%size;
            Q[rear] = value;
            c++;
      }
      void delete(){
            if(c==0){
                  System.out.println("Q is empty");
                  return;
            }
            System.out.println("Deleted item is: "+Q[front]);
            if(front==rear)
                  front=rear=-1;
            else
                  front = (front+1)%size;
            c--;
      }
      void display(){
            if(c==0){
                  System.out.println("Q is empty");
                  return;
            }
            int i=front;
            if(front<=rear){
                  while(i<=rear)
                        System.out.print(Q[i++]+" ");
            }
            else{
                  while(i!=rear){
                        System.out.print(Q[i]+" ");
                        i=(i+1)%size;
                  }
                  System.out.print(Q[i]);
            }
            System.out.println();
      }
}

class Test
{
      public static void main(String[] args)
      {
            CQArrays q = new CQArrays();
            q.insert(111);
            q.insert(222);
            q.insert(333);
            q.insert(444);
            q.insert(555);
            q.display();
            q.insert(666);//
            q.delete();//111
            q.display();
            q.insert(666);
            q.display();
      }
}
```

```
C:\prakashclasses>javac Test.java

C:\prakashclasses>java Test
111 222 333 444 555
Q is full
Deleted item is: 111
222 333 444 555
222 333 444 555 666
```

circular queue implementation by using linked list

```
class CQList
{
      Node front,rear;
      class Node
      {
            int data;
            Node next;
            Node(int data,Node next){
                  this.data = data;
                  this.next = next;
            }
```

```
    }
    void insert(int data){
        Node newNode = new Node(data,null);
        if(front==null)
            front = newNode;
        else
            rear.next = newNode;
        rear=newNode;
        rear.next=front;
    }
    void delete(){
        if(front==null){
            System.out.println("q is empty");
            return;
        }
        System.out.println("Deleted item is: "+front.data);
        if(front==rear){
            front=null;
            rear=null;
        }
        else{
            front = front.next;
            rear.next = front;
        }
    }
    void display(){
        Node temp = front;
        if(temp==null){
            System.out.println("q is empty");
            return;
        }
        while(temp.next!=front){
            System.out.print(temp.data+" ");
            temp=temp.next;
        }
        System.out.println(temp.data);
    }
}
class Test
{
    public static void main(String[] args)
    {
        CQList q = new CQList();
        q.display();
        q.insert(111);
        q.insert(222);
        q.insert(333);
        q.insert(444);
        q.insert(555);
        q.display();
        q.delete();
        q.delete();
        q.display();
    }
```

```
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
q is empty
111 222 333 444 555
Deleted item is: 111
Deleted item is: 222
333 444 555
```

deque:

normal queue -----> insertion is from rear and deletion is from front side

circular queue ---> insertion is from rear and deletion is from front side

deque ------------> we can perform insertion and deletion from both sides

deque implementation by using arrays:

```
class DQArrays
{
      int DQ[],front,rear,size;
      DQArrays(){
            front = -1;
            rear = -1;
            size = 5;
            DQ = new int[size];
      }
      void insertAtFront(int value){
            if((front==0 && rear==size-1)||(front==rear+1)){
                  System.out.println("Q is full");
                  return;
            }
            if(front==-1)
                  front=rear=0;
            else if(front==0)
                  front=size-1;
            else
                  front=front-1;
            DQ[front]=value;
      }
      void insertAtRear(int value){
            if((front==0 && rear==size-1)||(front==rear+1)){
                  System.out.println("Q is full");
                  return;
            }
            if(front==-1)
                  front=rear=0;
            else if(rear==size-1)
                  rear=0;
            else
                  rear=rear+1;
```

```
        DQ[rear]=value;
    }
    void deleteFromFront(){
        if(front==-1){
            System.out.println("DQ is empty");
            return;
        }
        System.out.println("deleted item is: "+DQ[front]);
        if(front==rear)
            front=rear=-1;
        else{
            if(front==size-1)
                front=0;
            else
                front=front+1;
        }
    }
    void deleteFromRear(){
        if(front==-1){
            System.out.println("dq is empty");
            return;
        }
        System.out.println("Deleted object : "+DQ[rear]);
        if(front==rear)
            front=rear=-1;
        else{
            if(rear==0)
                rear=size-1;
            else
                rear=rear-1;
        }
    }
    void display(){
        if(front==-1){
            System.out.println("dq is empty");
            return;
        }
        int left=front,right=rear;
        if(left<=right){
            while(left<=right)
                System.out.print(DQ[left++]+" ");
        }
        else{
            while(left<=size-1)
                System.out.print(DQ[left++]+" ");
            left=0;
            while(left<=right)
                System.out.print(DQ[left++]+" ");
        }
        System.out.println();
    }
}
class Test
{
    public static void main(String[] args)
    {
        DQArrays dq = new DQArrays();
        dq.insertAtRear(111);
        dq.insertAtRear(222);
        dq.insertAtRear(333);
        dq.display();//111 222 333
        dq.insertAtFront(777);
        dq.insertAtFront(888);
        dq.insertAtFront(999);//Q is full
        dq.display();//888 777 111 222 333
        dq.deleteFromFront();//888
        dq.display();//777 111 222 333
        dq.deleteFromRear();//333
        dq.display();//777 111 222
    }
}

deque implementation by using linked list
-----------------------------------------
import java.util.*;
class DequeList{
    Node front,rear;
    int size;//number of element in deque
    DequeList(){
        front = null;
        rear = null;
        size = 0;
    }
    class Node{
        int data;
        Node next;
        Node(int data,Node next){
            this.data = data;
            this.next = next;
            size++;
        }
    }
    void insertAtFront(int value){
        Node newNode = new Node(value,null);
        if(front==null){
            front = newNode;
            rear = newNode;
            return;
        }
        newNode.next = front;
        front = newNode;
    }
    void insertAtRear(int value){
        Node newNode = new Node(value,null);
        if(front==null){
            front = newNode;
            rear = newNode;
            return;
        }
        rear.next = newNode;
        rear = newNode;
    }
    void deleteAtFront(){
        if(front==null){
            System.out.println("dq is empty");
            return;
        }
        System.out.println("deleted obj is: "+front.data);
        front = front.next;
        size--;
    }
    void deleteAtRear(){
        if(front==null){
            System.out.println("dq is empty");
            return;
        }
        System.out.println("deleted obj is: "+rear.data);
        size--;
        if(front==rear){
            front=null;
            rear=null;
            return;
        }
        Node temp = front;
        while(temp.next!=rear)
            temp = temp.next;
        rear = temp;
        rear.next=null;
    }
    void display(){
        if(front==null){
            System.out.println("dq is empty");
            return;
        }
        Node temp = front;
        while(temp!=null){
            System.out.print(temp.data+" ");
            temp=temp.next;
        }
        System.out.println();
    }
}
class Test
{
    public static void main(String[] args)
    {
        DequeList dq = new DequeList();
        dq.insertAtFront(333);
        dq.insertAtFront(222);
        dq.insertAtFront(111);
        dq.display();//111 222 333
        dq.insertAtRear(444);
```

```
            dq.insertAtRear(555);
            dq.insertAtFront(999);
            dq.display();//999 111 222 333 444 555
            dq.deleteAtFront();
            dq.display();//111 222 333 444 555
            dq.deleteAtRear();
            dq.display();//111 222 333 444
      }
}
```

output:

```
111 222 333
999 111 222 333 444 555
deleted obj is: 999
111 222 333 444 555
deleted obj is: 555
111 222 333 444
```

predefined queue implementation:

```
Collection(I) -----> Queue(I)

Queue(I) ----------> PriorityQueue(C)
Queue(I) ----------> BlockingQueue(I)
Queue(I) ----------> Deque(I)
```

if we want to represent a group of individual object prior to processing, then we should go for Queue concept. it is child interface to Collection.

Queue(I):

=> 1.5 version of java

=> child class to Collection

=> not index based

=> allow duplicate object

=> allow all the elements prior to processing

=> FIFO

=> only homogeneous (same type)

=> null values are not allowed

=> only comparable objects are allowed

=> if you want non-comparable objects, then we should go for java.util.Comparator

boolean offer(object) ===> add an object into queue

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Queue q = new PriorityQueue();
            System.out.println(q);//[]
            q.offer(111);
            q.offer(222);
            q.offer(333);
            System.out.println(q);//[111,222,333]
      }
}
```

```
Object peek()     -----> return head/first element.
Object element() -----> return head/first element.
```

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Queue q = new PriorityQueue();
            System.out.println(q);//[]
            q.offer(111);
            q.offer(222);
            q.offer(333);
            q.offer(444);
            System.out.println(q);//[111,222,333,444]
            System.out.println(q.peek());//111
            System.out.println(q.element());//111
      }
}
```

What is the difference between peek() and element()?

On empty queue peek() method will return null, but element() method return exception saying "NoSuchElementException".

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Queue q = new PriorityQueue();
            System.out.println(q);//[]
            System.out.println(q.peek());//null
            System.out.println(q.element());//RE: NoSuchElementException
      }
}
```

```
Object poll() ----> remove and return head/first element
Object remove() --> remove and return head/first element
```

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Queue q = new PriorityQueue();
            System.out.println(q);//[]
            q.offer(111);
            q.offer(222);
            q.offer(333);
            q.offer(444);
            System.out.println(q);//[111,222,333,444]
            System.out.println(q.poll());//111
            System.out.println(q);//[222,333,444]
            System.out.println(q.remove());//222
            System.out.println(q);//[333,444]
      }
}
```

What is the difference between poll() and remove()?

On empty queue poll() method will return null, but remove() method return exception saying "NoSuchElementException".

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Queue q = new PriorityQueue();
            System.out.println(q);//[]
            System.out.println(q.poll());//null
            System.out.println(q.remove());//RE: NoSuchElementException
      }
}
```

Deque:
------
1) it is introduced in 1.6 version.
2) child interface to Queue interface.
3) Deque means double ended queue.
4) Deque allows insertion and deletions from both ends.

```
void addFirst(obj)  ------> add object at first
boolean offerFirst(obj) --> add object at first
```

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            dq.addFirst(111);
            dq.addFirst(222);
            System.out.println(dq);//[222,111]
            dq.offerFirst(333);
            dq.offerFirst(444);
            System.out.println(dq);//[444,333,222,111]
      }
}

void addLast(obj) -------> add object at end
boolean offerLast(obj) --> add object at end

Ex:
---
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            dq.addFirst(111);
            dq.addFirst(222);
            System.out.println(dq);//[222,111]
            dq.offerFirst(333);
            dq.offerFirst(444);
            System.out.println(dq);//[444,333,222,111]
            dq.addLast(999);
            System.out.println(dq);//[444,333,222,111,999]
            dq.offerLast(888);
            System.out.println(dq);//[444,333,222,111,999,888]
      }
}


Object getFirst() ----> get the first element from dq
Object peekFirst() ---> get the first element from dq

Ex:
---
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            dq.addFirst(111);
            dq.addFirst(222);
            dq.offerFirst(333);
            dq.offerFirst(444);
            dq.addLast(999);
            dq.offerLast(888);
            System.out.println(dq);//[444,333,222,111,999,888]
            System.out.println(dq.getFirst());//444
            System.out.println(dq.peekFirst());//444
      }
}
```

What is the difference between getFirst() and peekFirst()?

On empty deque object, if we call peekFirst() returns null, but if we call getFirst() method it raises exception saying "NoSuchElementException"

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            System.out.println(dq.peekFirst());//null
            System.out.println(dq.getFirst());//RE:
NoSuchElementException
      }
}
```

Object removeFirst() -----> remove first element from dq
Object pollFirst() -------> remove first element from dq

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            dq.addFirst(111);
            dq.addFirst(222);
            dq.offerFirst(333);
            dq.offerFirst(444);
            dq.addLast(999);
            dq.offerLast(888);
            System.out.println(dq);//[444,333,222,111,999,888]
            System.out.println(dq.removeFirst());//444
            System.out.println(dq);//[333,222,111,999,888]
            System.out.println(dq.pollFirst());//333
            System.out.println(dq);//[222,111,999,888]
      }
}
```

What is the difference between removeFirst() and PollFirst()?

On empty deque object, if we call pollFirst() returns null, but if we call removeFirst() method it raises exception saying "NoSuchElementException"

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            System.out.println(dq.pollFirst());//null
            System.out.println(dq.removeFirst());//RE:
NoSuchElementException
      }
}
```

Object removeLast() -----> remove last object from dq.
Object pollLast() -------> remove last object from dq

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            dq.addFirst(111);
            dq.addFirst(222);
            dq.offerFirst(333);
            dq.offerFirst(444);
            dq.addLast(999);
            dq.offerLast(888);
            System.out.println(dq);//[444,333,222,111,999,888]
            System.out.println(dq.removeLast());//888
            System.out.println(dq);//[444,333,222,111,999]
            System.out.println(dq.pollLast());//999
            System.out.println(dq);//[444,333,222,111]
      }
}
```

What is the difference between removeLast() and PollLast()?

On empty deque object, if we call pollLast() returns null, but if we call removeLast() method it raises exception saying "NoSuchElementException"

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            System.out.println(dq.pollLast());//null
            System.out.println(dq.removeLast());//RE: 
NoSuchElementException
      }
}
```

```
boolean removeFirstOccurrence(object) ----> remove first occurrence
boolean removeLastOccurrence(object)  ----> remove last occurrence
```

Ex:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Deque dq = new ArrayDeque();
            System.out.println(dq);//[]
            dq.addLast(111);
            dq.addLast(222);
            dq.addLast(333);
            dq.addLast(111);
            dq.addLast(222);
            dq.addLast(111);
            dq.addLast(444);
            dq.addLast(555);
            System.out.println(dq);//[111,222,333,111,222,111,444,555]
            dq.removeFirstOccurrence(111);
            System.out.println(dq);//[222,333,111,222,111,444,555]
            dq.removeLastOccurrence(111);
            System.out.println(dq);//[222,333,111,222,444,555]
      }
}
```

Bit manipulations:
~~~~~~~~~~~~~~~~~~

01. Introduction to number systems
02. Types of number systems
03. Decimal to Binary conversion
04. Binary to Decimal conversion
05. Bitwise operators
06. Even or Odd number application
07. Swaping of two numbers application
08. Bit level operations (get, set, clear and update)
09. Clear last 'i' bits
~~~~~~~~~~~~~~~~~~

10. Clear range of bits (i to j)
11. Number is power of two or not application
12. Count set bits applications
13. Fast exponetiation calculation application
14. Increment a value by one unit
15. Conversion from lower case string into upper case
16. Conversion from upper case string into lower case

Introduction to number systems:

A digital system can understand the digits by using the following components.

1) The digit
2) The position of the digit in the number
3) The base of the number system

Types of number systems:

The purpose of number systems are used to represent the number in digital systems.

1) Binary Number System
2) Decimal Number System
3) Octal Number System
4) Hexadecimal Number System

Binary Number System:

Digits : 0 and 1
Base : 2

Decimal Number System

Digits : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Base : 10

Octal Number System

Digits : 0, 1, 2, 3, 4, 5, 6, 7
Base : 8

Hexadecimal Number System

Digits : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f
Base : 16

Decimal Number	Binary Number	Octal Number	Hexadecimal Number
0	0000	0	0

1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	a
11	1011	13	b
12	1100	14	c
13	1101	15	d
14	1110	16	e
15	1111	17	f

Decimal to Binary Conversion:

Method1:

1) divide the given decimal number by '2', where it gives result along with remainder.
2) we have to store these remainders in a container.
3) we have to print the list values which are stored in reverse order.

Ex:

```
      13 ---->

      13/2 ---> 6 ----> 1
      6/2 ----> 3 ----> 0
      3/2 ----> 1 ----> 1
      1/2 ----> 0 ----> 1

      ans: 1101
```

Method2:

Find the binary equalent for the given number by using 8-4-2-1 code

```
Ex:
---
     15 ----> 1111
     13 ----> 1101

Binary to decimal conversion
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Method1:
--------
1) read the digits or symbols one by one from right to left.
2) multiply each bit with 2 power x where x = 0,1,2,3....
3) sum of these expression is the decimal number

Ex:
     1010

     0x2^0 = 0
     1x2^1 = 2
     0x2^2 = 0
     1x2^3 = 8
     ---------
            10
     ---------

Method2:
--------
By using 8-4-2-1 code

Ex:
     1010 ----> 8+2=10
     1011 ----> 8+2+1=11

Bitwise operators
~~~~~~~~~~~~~~~~~
Bitwise operators are the special operators in almost all the programming
languages. These operators are faster and an efficient way to interact
with computers to make heavy computation in a linear time because it
works directly with bits rather than a level of conversion internally.
The following are the various bitwise operators supported by all most all
the programming languages.

1) bitwise and &
2) bitwise or |
3) bitwise x-or ^
4) bitwise left shift <<
5) bitwise right sift >>
6) bitwise complement ~

bitwise and &
-------------
it returns 1 if both bits are 1 else 0

truth table
```

```
-----------
0 & 0 = 0
1 & 0 = 0
0 & 1 = 0
1 & 1 = 1
```

Ex:
---

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            int a = 5, b = 9;
            System.out.println(a&b);//1
      }
}
```

bitwise or |
----

it returns 1 if any one bit 1 else 0

truth table

```
0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1
```

Ex:
---

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            int a = 5, b = 9;
            System.out.println(a|b);//13
      }
}
```

bitwise x-or ^
----

it returns 1 if both bits are in different state else returns 0

truth table

```
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0
```

Ex:
---

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            int a = 5, b = 9;
            System.out.println(a^b);//12
      }
}

bitwise shift operators:
~~~~~~~~~~~~~~~~~~~~~~~~
shift operators are used to shift the bits of a number to left or right
direction.

Ex:
---
      n=5
      n ------> 5
      n<<1 ---> 5*2^1 = 5*2=10
      n<<2 ---> 5*2^2 = 5*4=20

a,b ---> a<<b = a*2^b

Ex:
---
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            int a = 5;
            System.out.println(a);//5
            System.out.println(a<<1);//10
            System.out.println(a<<2);//20
            System.out.println(a<<3);//40
      }
}


Ex:
---
      n=5
      n ------> 5
      n>>1 ---> 5/2^1 = 5/2=2
      n>>2 ---> 5/2^2 = 5/4=1

a,b ---> a>>b = a/2^b

Ex:
---
import java.util.*;
class Test
{
```

```
    public static void main(String[] args)
    {
        int a = 5;
        System.out.println(a);//5
        System.out.println(a>>1);//2
        System.out.println(a>>2);//1
        System.out.println(a>>3);//0
    }
}
```

bitwise complement:

It is represented as ~, i.e. all the bits are inverted, every 0 as 1 and 1 as 0.

formula: ~n = -(n+1)

Ex:

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        System.out.println(~5);//-(5+1)=-6
        System.out.println(~7);//-(7+1)=-8
        System.out.println(~1);//-(1+1)=-2
        System.out.println(~-2);//-(-2+1)=-(-1)=1
    }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
-6
-8
-2
1
```

Advantages of bitwise operators

1) speed
2) space optimization
3) bit manipulation
4) code simplification
5) readability will be improved
6) data encryption

even or odd number application:

0	0000
1	0001
2	0010

3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010

if LBS is 0 then the given number is even
if LSB is 1 then the given number is odd

bitMask = 1

if (n&bitMask)==0 then EVEN else ODD

Ex:

```
import java.util.*;
class Test
{
    static String evenOrOdd(int n){
        int bitMask = 1;
        if((n&bitMask)==0)
            return "Even Number";
        else
            return "Odd Number";
    }
    public static void main(String[] args)
    {
        for(int i=0;i<=10;i++){
            System.out.println(i+"\t"+evenOrOdd(i));
        }
    }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
0       Even Number
1       Odd Number
2       Even Number
3       Odd Number
4       Even Number
5       Odd Number
6       Even Number
7       Odd Number
8       Even Number
9       Odd Number
10      Even Number
```

swaping of two numbers:

```
import java.util.*;
class Test
{
      public static void main(String[] args)
      {
            Scanner obj = new Scanner(System.in);
            System.out.println("Enter a value:");
            int a = obj.nextInt();
            System.out.println("Enter b value:");
            int b = obj.nextInt();

            System.out.println("Before swaping   : a= "+a+" and b= "+b);
            a = a^b;
            b = a^b;
            a = a^b;
            System.out.println("After  swaping   : a= "+a+" and b= "+b);
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
Enter a value:
12
Enter b value:
33
Before swaping   : a= 12 and b= 33
After  swaping   : a= 33 and b= 12
```

```
bit level operations:
---------------------
getting ith bit from a binary number
setting ith bit in a binary number
clearing ith bit in a binary number
updating ith bit in a binary number

get ith bit:
~~~~~~~~~~~~

Ex:
---
      10 -----> 1010

      0th -----> 0
      1st -----> 1
      2nd -----> 0
      3rd -----> 1

bitMask = 1<<i

if n & bitMask == 0 then 0 else 1

Ex:
```

```
import java.util.*;
class Test
{
      public static int getIthBit(int n,int i){
            int bitMask = 1<<i;
            if((n & bitMask) == 0)
                  return 0;
            else
                  return 1;
      }
      public static void main(String[] args)
      {
            //19 ---> 10011
            System.out.println(getIthBit(19,0));//1
            System.out.println(getIthBit(19,1));//1
            System.out.println(getIthBit(19,2));//0
            System.out.println(getIthBit(19,3));//0
            System.out.println(getIthBit(19,4));//1
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
1
1
0
0
1
```

set ith Bit:

Ex:

```
      19        ----> 10011

      set 0th   ----> 10010 ---> 18
      set 1st ------> 10001 ---> 17
      set 2nd ------> 10111 ---> 23
      set 3rd ------> 11011 ---> 27
      set 4th ------> 00011 ---> 3
```

bitmask = 1<<i

formula: n ^ bitmask

Ex:

```
import java.util.*;
class Test
{
      public static int getIthBit(int n,int i){
            int bitMask = 1<<i;
            if((n & bitMask) == 0)
                  return 0;
            else
                  return 1;
      }
      public static int setIthBit(int n,int i){
            int bitMask = 1<<i;
            return n ^ bitMask;
      }
      public static void main(String[] args)
      {
            //19 ---> 10011
            System.out.println(setIthBit(19,0));//18
            System.out.println(setIthBit(19,1));//17
            System.out.println(setIthBit(19,2));//23
            System.out.println(setIthBit(19,3));//27
            System.out.println(setIthBit(19,4));//3
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
18
17
23
27
3
```

clear ith bit:

bitMask = ~(1<<i)

formula : n & bitMask

Ex:

```
import java.util.*;
class Test
{
      public static int getIthBit(int n,int i){
            int bitMask = 1<<i;
            if((n & bitMask) == 0)
                  return 0;
            else
                  return 1;
      }
      public static int setIthBit(int n,int i){
            int bitMask = 1<<i;
            return n ^ bitMask;
      }
      public static int clearIthBit(int n,int i){
            int bitMask = ~(1<<i);
            return n&bitMask;
      }
      public static void main(String[] args)
      {
            //19 ---> 10011
            System.out.println(clearIthBit(19,0));//10011=>10010=>18
            System.out.println(clearIthBit(19,1));//10011=>10001=>17
            System.out.println(clearIthBit(19,2));//10011=>10011=>19
            System.out.println(clearIthBit(19,3));//10011=>10011=>19
            System.out.println(clearIthBit(19,4));//10011=>00011=>3
      }
}

update ith bit:
---------------
import java.util.*;
class Test
{
      public static int getIthBit(int n,int i){
            int bitMask = 1<<i;
            if((n & bitMask) == 0)
                  return 0;
            else
                  return 1;
      }
      public static int setIthBit(int n,int i){
            int bitMask = 1<<i;
            return n ^ bitMask;
      }
      public static int clearIthBit(int n,int i){
            int bitMask = ~(1<<i);
            return n&bitMask;
      }
      public static int updateIthBit(int n,int i,int nb){
            if(nb==0)
                  return clearIthBit(n,i);
            else
                  return n;
      }

      public static void main(String[] args)
      {
            //19 ---> 10011
            System.out.println(updateIthBit(19,0,0));//10011=>10010=>18
            System.out.println(updateIthBit(19,0,1));//10011=>10011=>19
            System.out.println(updateIthBit(19,1,0));//10011=>10001=>17
            System.out.println(updateIthBit(19,1,1));//10011=>10011=>19
      }
}

clear last i bits:
------------------
```

```
19 -----> 10011

clear 1 -----> 10010
clear 2 -----> 10000
clear 3 -----> 10000
clear 4 -----> 10000
clear 5 -----> 00000


bitmask =(-1) << i

formula: n & bitmask

Ex:
---
import java.util.*;
class Test
{
	public static int clearLastIBits(int n,int i){
		int bitMask = (-1)<<i;
		return n & bitMask;
	}
	public static void main(String[] args)
	{
		//19 ---> 10011
		System.out.println(clearLastIBits(19,1));//10011=>10010=>18
		System.out.println(clearLastIBits(19,2));//10011=>10000=>16
		System.out.println(clearLastIBits(19,3));//10011=>10000=>16
		System.out.println(clearLastIBits(19,4));//10011=>10000=>16
		System.out.println(clearLastIBits(19,5));//10011=>00000=>0
	}
}

number is power of two or not application
------------------------------------------
import java.util.*;
class Test
{
	public static boolean powerOf2(int n){
		return (n & (n-1))==0;
	}
	public static void main(String[] args)
	{
		for(int i=0;i<=10;i++)
			System.out.println(i+"\t"+powerOf2(i));
	}
}

C:\prakash>javac Test.java

C:\prakash>java Test
0       true
1       true
2       true
```

```
3       false
4       true
5       false
6       false
7       false
8       true
9       false
10      false
```

count set bits in a number

```
import java.util.*;
class Test
{
	public static int countSetBits(int n){
		int c=0;
		while(n!=0){
			if((n&1)!=0)
				c++;
			n=n>>1;
		}
		return c;
	}
	public static void main(String[] args)
	{
		for(int i=0;i<=10;i++)
			System.out.println(i+"\t"+countSetBits(i));
	}
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
0       0
1       1
2       1
3       2
4       1
5       2
6       2
7       3
8       1
9       2
10      2
```

increment a number by one unit

```
import java.util.*;
class Test
{
	public static int increment(int n){
		return -~n;
	}
```

```
      public static void main(String[] args)
      {
            for(int i=0;i<=10;i++)
                  System.out.println(i+"\t"+increment(i));
      }
}

C:\prakash>javac Test.java

C:\prakash>java Test
0       1
1       2
2       3
3       4
4       5
5       6
6       7
7       8
8       9
9       10
10      11

lower case to upper case conversion:
------------------------------------
import java.util.*;
class Test
{
      public static String convertToUpperCase(String s){
            String ss="";
            for(int i=0;i<s.length();i++){
                  ss=ss+(char)(s.charAt(i)^32);
            }
            return ss;
      }
      public static void main(String[] args)
      {
            System.out.println(convertToUpperCase("abc"));
      }
}

C:\prakash>javac Test.java

C:\prakash>java Test
ABC

upper case to lower case conversion
-----------------------------------
import java.util.*;
class Test
{
      public static String convertToLowerCase(String s){
            String ss="";
            for(int i=0;i<s.length();i++){
                  ss=ss+(char)(s.charAt(i)|32);
            }
            return ss;
      }
      public static void main(String[] args)
      {
            System.out.println(convertToLowerCase("ABC"));
      }
}
```

```
Fast exponetiation calculation application
------------------------------------------
import java.util.*;
class Test
{
      public static int fastExpo(int a,int n){
            int res=1;
            while(n!=0){
                  if((n&1)!=0)
                        res = res * a;
                  a = a * a;
                  n = n>>1;
            }
            return res;
      }

      public static void main(String[] args)
      {
            System.out.println(fastExpo(2,8));//256
      }
}
```

```
Hashtable data structure:
~~~~~~~~~~~~~~~~~~~~~~~~~
01. introduction
02. hashtable
03. operations
04. hash function
05. implementation of hash table
06. collisions
07. collection resolution methods
08. linear probing & implementation
09. quadratic probing & implementation
10. separate chaining & implementation
```

```
introduction:
~~~~~~~~~~~~~
```

In the case of searching algorithms, consider the problem of searching for a value in an array. if the array is not sorted then we have to compare the given key value with all elements one-by-one, it will take time complexity O(n), if the array is sorted then time complexity is O(logn).

linear ----> O(n)

binary ----> O(logn)

O(n)>O(logn)>O(1)

it is possible to get the location of given key by using some magic method. almost it takes O(1) i.e. constant time. hash method or hash function works just like this magic method.

hashtable
~~~~~~~~~
=> hashtable is a data structure that maps keys to the values.
=> each position of the hashtable is called as slot or bucket.
=> the hashtable uses hash function to calculate an index of a value.(insert/delete/search)

the process of storing data using hash function in a hash table as follows

1) create an array of size 'N'. this array is called as hashtable.
2) find the hashcode of the given data by using hash function.
3) take modulo of hash code with table size to get index of array to store the data.
4) finally store this data in the position that we calculated.

operations
~~~~~~~~~~
1) insertion
2) deletion
3) search for data

hash function
~~~~~~~~~~~~~
a hash function or hash method is a function/method that generate index in a table for the given key value/object/ an ideal hash function generates a unique hash value for every value.

implementation of hash table
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
import java.util.*;
class Hashtable{
      int size;
      int a[];
      Hashtable(int size){
            this.size = size;
            a = new int[this.size];
            for(int i=0;i<this.size;i++)
                  a[i] = -1;
      }
      void print(){
            System.out.println("content of hash table:");
            for(int i=0;i<size;i++)
                  System.out.println(i+" ====> "+a[i]);
      }
      int compute(int value){
```

```
            return value%size;
      }
      boolean add(int value){
            int hcode = compute(value);
            if(a[hcode]==-1){
                  a[hcode]=value;
                  return true;
            }
            return false;
      }
      boolean delete(int value){
            int hcode = compute(value);
            if(a[hcode]!=-1 && a[hcode]==value){
                  a[hcode] = -1;
                  return true;
            }
            return false;
      }
      boolean search(int value){
            int hcode = compute(value);
            if(a[hcode]==value)
                  return true;
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Hashtable h = new Hashtable(10);
            h.print();
            System.out.println(h.add(23));
            System.out.println(h.add(24));
            System.out.println(h.add(33));
            System.out.println(h.add(50));
            System.out.println(h.add(105));
            System.out.println(h.add(177));
            System.out.println(h.add(777));
            System.out.println(h.add(999));
            h.print();
            System.out.println(h.delete(33));//false
            System.out.println(h.delete(23));//true
            System.out.println(h.delete(100));//true
            h.print();
            System.out.println(h.search(23));//false
            System.out.println(h.search(999));//true
            System.out.println(h.search(100));//false
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
content of hash table:
```

```
0 ====> -1
1 ====> -1
2 ====> -1
3 ====> -1
4 ====> -1
5 ====> -1
6 ====> -1
7 ====> -1
8 ====> -1
9 ====> -1
true
true
false
true
true
true
false
true
content of hash table:
0 ====> 50
1 ====> -1
2 ====> -1
3 ====> 23
4 ====> 24
5 ====> 105
6 ====> -1
7 ====> 177
8 ====> -1
9 ====> 999
false
true
false
content of hash table:
0 ====> 50
1 ====> -1
2 ====> -1
3 ====> -1
4 ====> 24
5 ====> 105
6 ====> -1
7 ====> 177
8 ====> -1
9 ====> 999
false
true
false
```

06. collisions

~~~~~~~~~~~~~~

When a hash function generates the same index/hcode for the two or more different keys, this problem is called as collision. a hash function should return unique address for each key. but practically it is not possible.
~~~~~~~~~~~~~~

properties of a good hash function:

1) it should provide a uniform distribution of hash values.
2) choose a hash function, which can be computed quickly and returns unique id.
3) choose a hash function, with a good collision resolution algorithms.
4) choose a hash function, which uses the nessary info provided in the key.
5) it should have a high load factor for given set of keys.

StringBuffer ----> 16 => 34
Hashtable -------> 10 => 20

collision resolution methods

Hash collisions are practically unavoidable when hashing large number of values. the following are the techniques that are used to find the alternative location in the hash table for the given objects.

1) linear probing
2) quadratic probing
3) seperate chaining

linear probing & implementation

Insert 3, 13, 23, 33, 43, 53, 63

```
boolean add(int value){
      int hcode = compute(value);
      for(int i=0;i<size;i++){
            if(a[hcode]==-1){
                  a[hcode]=value;
                  return true;
            }
            hcode = hcode + compute1(i);
            hcode = hcode % size;
      }
      return false;
}
```

There is no proper place for inserting 63 in the hashtable

```
import java.util.*;
class Hashtable{
      int size;
      int a[];
      Hashtable(int size){
            this.size = size;
            a = new int[this.size];
            for(int i=0;i<this.size;i++)
                  a[i] = -1;
      }
      void print(){
          System.out.println("content of hash table:");
            for(int i=0;i<size;i++)
                System.out.println(i+" ====> "+a[i]);
      }
      int compute(int value){
            return value%size;
      }
      int compute1(int index){
            return index;//linear probing
      }
      boolean add(int value){
            int hcode = compute(value);
            for(int i=0;i<size;i++){
                if(a[hcode]==-1){
                       a[hcode]=value;
                       return true;
                }
                hcode = hcode + compute1(i);
                hcode = hcode % size;
            }
            return false;
      }
      boolean delete(int value){
            int hcode = compute(value);
            for(int i=0;i<size;i++){
                if(a[hcode]!=-1 && a[hcode]==value){
                       a[hcode]=-1;
                       return true;
                }
                hcode = hcode + compute1(i);
                hcode = hcode % size;
            }
            return false;
      }
      boolean search(int value){
            int hcode = compute(value);
            for(int i=0;i<size;i++){
                if(a[hcode]==value)
                return true;
                hcode = hcode + compute1(i);
                hcode = hcode % size;
            }
            return false;
      }
  }
  class Test
  {
      public static void main(String[] args)
      {
            Hashtable h = new Hashtable(10);
            System.out.println(h.add(3));
            System.out.println(h.add(13));
            System.out.println(h.add(23));
            System.out.println(h.add(33));
```

```
            System.out.println(h.add(43));
            System.out.println(h.add(53));
            System.out.println(h.add(63));
            h.print();
            System.out.println(h.search(43));//true
            System.out.println(h.search(93));//false
            h.delete(23);
            h.print();
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
true
true
true
true
true
true
false
content of hash table:
0 ====> -1
1 ====> 53
2 ====> -1
3 ====> 3
4 ====> 13
5 ====> -1
6 ====> 23
7 ====> -1
8 ====> 43
9 ====> 33
true
false
content of hash table:
0 ====> -1
1 ====> 53
2 ====> -1
3 ====> 3
4 ====> 13
5 ====> -1
6 ====> -1
7 ====> -1
8 ====> 43
9 ====> 33
```

```
quadratic probing & implementation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

linear ----> compute1(index) ---> index
quadratic--> compute2(index) ---> index*index


objects: 5, 15, 25
```

```
import java.util.*;
class Hashtable{
      int size;
      int a[];
      Hashtable(int size){
            this.size = size;
            a = new int[this.size];
            for(int i=0;i<this.size;i++)
                  a[i] = -1;
      }
      void print(){
            System.out.println("content of hash table:");
            for(int i=0;i<size;i++)
                  System.out.println(i+" ====> "+a[i]);
      }
      int compute(int value){
            return value%size;
      }
      int compute2(int index){
            return index*index;//quadratic probing
      }
      boolean add(int value){
            int hcode = compute(value);
            for(int i=0;i<size;i++){
                  if(a[hcode]==-1){
                        a[hcode]=value;
                        return true;
                  }
                  hcode = hcode + compute2(i);
                  hcode = hcode % size;
            }
            return false;
      }
      boolean delete(int value){
            int hcode = compute(value);
            for(int i=0;i<size;i++){
                  if(a[hcode]!=-1 && a[hcode]==value){
                        a[hcode]=-1;
                        return true;
                  }
                  hcode = hcode + compute2(i);
                  hcode = hcode % size;
            }
            return false;
      }
      boolean search(int value){
            int hcode = compute(value);
            for(int i=0;i<size;i++){
                  if(a[hcode]==value)
                  return true;
                  hcode = hcode + compute2(i);
                  hcode = hcode % size;
            }
```

```
        return false;
    }
}
class Test
{
    public static void main(String[] args)
    {
        Hashtable h = new Hashtable(10);
        h.add(5);
        h.add(15);
        h.add(25);
        h.add(35);
        h.add(45);
        h.print();
        System.out.println(h.search(15));//true
        System.out.println(h.search(35));//true
        System.out.println(h.search(45));//false
        h.delete(15);
        h.print();
    }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
content of hash table:
0 ====> 25
1 ====> -1
2 ====> -1
3 ====> -1
4 ====> -1
5 ====> 5
6 ====> 15
7 ====> -1
8 ====> -1
9 ====> 35
true
true
false
content of hash table:
0 ====> 25
1 ====> -1
2 ====> -1
3 ====> -1
4 ====> -1
5 ====> 5
6 ====> -1
7 ====> -1
8 ====> -1
9 ====> 35
```

```
separate chaining & implementation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

In collision handling method chaining is a concept which introduces an additional filed data. i.e. chain. A seperate chain table is maintained for colliding objects, when collision occurrs then a linked list (chain) is maintained at the home bucket.

```
import java.util.*;
class Hashtable{
      int size;
      Node a[];
      class Node{
            int value;
            Node next;
            Node(int value,Node next){
                  this.value = value;
                  this.next = next;
            }
      }
      Hashtable(int size){
            this.size = size;
            a = new Node[this.size];
            for(int i=0;i<this.size;i++)
                  a[i] = null;
      }
      void print(){
            System.out.println("content of hash table:");
            for(int i=0;i<size;i++)
            {
                  Node head = a[i];
                  while(head!=null){
                        System.out.print(head.value+" => ");
                        head = head.next;
                  }
                  System.out.println("null");
            }
      }
      int compute(int value){
            return value%size;
      }
      void add(int value){
            int hcode = compute(value);
            a[hcode] = new Node(value,a[hcode]);
      }
      boolean delete(int value){
            int hcode = compute(value);
            Node nextNode, head = a[hcode];
            if(head!=null && head.value==value){
                  a[hcode] = head.next;
                  return true;
            }
            while(head!=null){
                  nextNode = head.next;
                  if(nextNode!=null && nextNode.value==value){
                        head.next = nextNode.next;
                        return true;
                  }
                  else
                        head = nextNode;
            }
            return false;
      }
      boolean search(int value){
            int hcode = compute(value);
            Node head = a[hcode];
            while(head!=null){
                  if(head.value == value){
                        return true;
                  }
                  head = head.next;
            }
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Hashtable h = new Hashtable(10);
            h.add(13);
            h.add(14);
            h.add(15);
            h.add(16);
            h.add(19);
            h.add(23);
            h.add(33);
            h.add(43);
            h.add(333);
            h.print();
            System.out.println(h.search(333));//true
            System.out.println(h.search(999));//false
            System.out.println(h.delete(333));//true
            h.print();
      }
}
```

```
C:\prakash>javac Test.java

C:\prakash>java Test
content of hash table:
null
null
null
333 => 43 => 33 => 23 => 13 => null
14 => null
15 => null
16 => null
null
null
19 => null
```

```
true
false
true
content of hash table:
null
null
null
43 => 33 => 23 => 13 => null
14 => null
15 => null
16 => null
null
null
19 => null
```

Tree Data Structure:

introduction:

==> non-linear data structure.
==> best suitable for search operations.
==> hierarchical relationships (parent-child)

Ex: Banking application
Ex: Software
Ex: File System
Ex: Family Tree

A tree is a finate set of one or more nodes such that

i) There is a sepcially desinged node called ROOT.
ii) Remaining nodes are partitioned into n>=0 disjoint sets, called as sub-trees

Tree Terminologies

Root:

It is a unique node, which is not having incoming edges
Ex: A

Node:

Fundamental element of tree, each node has data and two pointers that may be point to null or its children.

Ex: A,B,C,D,E,F,G,H,I

Edge:

it is fundamental part of tree, used to connect two nodes

Ex: AB, AC, AD, BE, BF, BG, EI, DH

Path:
-----

an ordered list of nodes, that are connected by edges called as path

Ex: A to G ---> AB, BG

Leaf Nodes:
----

the nodes which is not having any children or outgoing edges

Ex: I, F, G, C, H

Height of tree:
----

height of tree is number of edges on the longest path between the root and leaf

Ex: AI--> 3

Level of tree:

The level of node is the number of edges on the path rom root to that node.

parent:

Node is a parent of all the child nodes that are linked by outgoing edges.

Ex: B,C,D,E

children:

Nodes that are having incoming edges and outgoing edges

Ex: B, C, D, E,

siblings:

Nodes in the tree what are children of the same parent is called as siblings

Ex: (BCD), (EFG)

ancestor:

A node reachanble through repeated moving from child to parent

Ex: I --> EBA

degree of node:

The total number of sub-trees attached to the node is called the degree of node.

Ex: De(B) = 3

degree of tree:

The max degree of all nodes is called as degree of the tree

Ex: de(tree) ---> 3

predecessor:

while displaying or traversing a tree, if a node occurrs previous to some other node, then that node is called as predecessor.

Ex: E is predecessor of I

sucessor:

while displaying or traversing a tree, if a node comes next to some other node is called as successor node.

Ex: E is sucessor of B

Binary Tree

A binary tree is a type of tree in which each node has at most two children (0, 1 or 2) which are referred to as the left child and right child.

In Binary tree each node will have one data filed and two pointers which is pointing to left sub-tree and right sub-tree. the degree of each node in the binary tree will be at the two.

Examples--> diagram

Properties of binary tree

1) max no of nodes on level i of a binary tree is 2^i.
2) there should be exactly one path should be there from root to any node.
3) tree with 'N' nodes has exactly 'N-1' edges connecting these nodes.
4) the height of complete binary tree of N nodes is logN

binary tree representation:

There are two ways are there to represent binary trees

1) sequential representation
2) linked list representation

Each node is sequentially arranged from to to bottom and from left to right let us understand this method, by numering each node the numberign will start from root node and then remaining nodes will given with increasing numbers in level wise direction if the nodes are in same level the numbers will be assiginged from left to right

```
left(n) = 2n+1
right(n) = 2n+2

n=0 ---> left(A) = 2n+1 = 1 ==> B
         right(A) = 2n+2 = 2 => C

n=2
left(C) = 2n+1 = 2x2+1=5 ----> F
right(C) = 2n+2 = 2x2+2=4+2=6 --> G
```

In linked list representation each node will be having three files

1) left pointer
2) data
3) right pointer

construction of binary tree using sequential representation:

=> pre order
=> post order
=> in order
=> level order

build tree by using preoder

Ex:

```
import java.util.*;
class BT{
      static int index=-1;
      class Node{
            int data;
            Node left;
            Node right;
            Node(int data){
                  this.data = data;
                  this.left = null;
                  this.right = null;
            }
      }
      Node buildTree(int[] nodes){
            index++;
            if(nodes[index]==-1)
                  return null;
            Node node = new Node(nodes[index]);
            node.left = buildTree(nodes);
            node.right = buildTree(nodes);
            return node;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int[] nodes = {1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1};
            BT obj = new BT();
            System.out.println(obj.buildTree(nodes).data);//1
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
1
```

Ex:

```
---
import java.util.*;
class BT{
      static int index=-1;
      class Node{
            int data;
            Node left;
            Node right;
            Node(int data){
                  this.data = data;
                  this.left = null;
                  this.right = null;
            }
      }
      Node buildTree(int[] nodes){
            index++;
            if(nodes[index]==-1)
                  return null;
            Node node = new Node(nodes[index]);
            node.left = buildTree(nodes);
            node.right = buildTree(nodes);
            return node;
      }
}
class Test
{
      public static void main(String[] args)
      {
            int[] nodes = {15,13,7,1,-1,-1,2,-1,-1,6,-1,-1,12,-1,-1};
            BT obj = new BT();
            //System.out.println(obj.buildTree(nodes).data);//15
            //System.out.println(obj.buildTree(nodes).left.data);//13
            System.out.println(obj.buildTree(nodes).right.data);//12
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
13

C:\test>javac Test.java

C:\test>java Test
12
```

binary tree traversals

Traversing the tree means visiting each node exactly one. basically there are three ways to traverse tree

1) Inorder (LDR)
2) Preorder (DLR)
3) Postorder (LRD)

Inorder traversal:

In this traversal the following are the rules to be followed

1) Left node
2) Root node
3) Right node

Preorder traversal:

In this traversal the following are the rules to be followed

1) Root node
2) Left node
3) Right node

Postoder traversal:

In this traversal the following are the rules to be followed

1) Left node
2) Right node
3) Root node

Ex:

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void preOrder(Node node){ //DLR
            if(node==null)
                  return;
            System.out.print(node.data+" ");
            preOrder(node.left);
            preOrder(node.right);
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      void postOrder(Node node){
            if(node==null)
                  return;
            postOrder(node.left);
            postOrder(node.right);
            System.out.print(node.data+" ");
      }
}
class Test
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            System.out.print("InOrder Traversal ===> ");
            obj.inOrder(obj.root);
            System.out.println();
            System.out.print("PreOrder Traversal ===> ");
            obj.preOrder(obj.root);
            System.out.println();
            System.out.print("PostOrder Traversal ===> ");
            obj.postOrder(obj.root);
            System.out.println();
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
InOrder Traversal ===> 2 1 3
PreOrder Traversal ===> 1 2 3
```

```
PostOrder Traversal ===> 2 3 1
```

```
Ex:
---
import java.util.*;
class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
class BT{
    Node root;
    BT(){
        root = null;
    }
    void preOrder(Node node){ //DLR
        if(node==null)
            return;
        System.out.print(node.data+" ");
        preOrder(node.left);
        preOrder(node.right);
    }
    void inOrder(Node node){
        if(node==null)
            return;
        inOrder(node.left);
        System.out.print(node.data+" ");
        inOrder(node.right);
    }
    void postOrder(Node node){
        if(node==null)
            return;
        postOrder(node.left);
        postOrder(node.right);
        System.out.print(node.data+" ");
    }
}
class Test
{
    public static void main(String[] args)
    {
        BT obj = new BT();
        obj.root = new Node(1);
        obj.root.left = new Node(2);
        obj.root.right = new Node(3);
        obj.root.left.left = new Node(4);
        obj.root.left.right = new Node(5);
        System.out.print("InOrder Traversal ===> ");
```

```
            obj.inOrder(obj.root);
            System.out.println();
            System.out.print("PreOrder Traversal ===> ");
            obj.preOrder(obj.root);
            System.out.println();
            System.out.print("PostOrder Traversal ===> ");
            obj.postOrder(obj.root);
            System.out.println();
      }
}

C:\test>javac Test.java

C:\test>java Test
InOrder Traversal ===> 4 2 5 1 3
PreOrder Traversal ===> 1 2 4 5 3
PostOrder Traversal ===> 4 5 2 3 1

level order traversal
---------------------
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void preOrder(Node node){ //DLR
            if(node==null)
                  return;
            System.out.print(node.data+" ");
            preOrder(node.left);
            preOrder(node.right);
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      void postOrder(Node node){
            if(node==null)
                  return;
            postOrder(node.left);
            postOrder(node.right);
            System.out.print(node.data+" ");
      }
      public static void levelOrderTraversal(Node node){
            if(node==null)
                  return;
            Queue<Node> q = new LinkedList<>();
            q.add(node);
            q.add(null);
            while(!q.isEmpty()){
                  Node cur = q.remove();
                  if(cur==null){
                        System.out.println();
                        if(q.isEmpty())
                              break;
                        else
                              q.add(null);
                  }
                  else{
                        System.out.print(cur.data+" ");
                        if(cur.left!=null)
                              q.add(cur.left);
                        if(cur.right!=null)
                              q.add(cur.right);
                  }
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            obj.root.left.left = new Node(4);
            obj.root.left.right = new Node(5);
            obj.root.right.left = new Node(6);
            obj.root.right.right = new Node(7);
            System.out.println("Level Order Traversal ");
            obj.levelOrderTraversal(obj.root);
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
Level Order Traversal
1
2 3
4 5 6 7
```

count nodes

```
~~~~~~~~~~~
import java.util.*;
class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
class BT{
    Node root;
    BT(){
        root = null;
    }
    void inOrder(Node node){
        if(node==null)
            return;
        inOrder(node.left);
        System.out.print(node.data+" ");
        inOrder(node.right);
    }
    public static void levelOrderTraversal(Node node){
        if(node==null)
            return;
        Queue<Node> q = new LinkedList<>();
        q.add(node);
        q.add(null);
        while(!q.isEmpty()){
            Node cur = q.remove();
            if(cur==null){
                System.out.println();
                if(q.isEmpty())
                    break;
                else
                    q.add(null);
            }
            else{
                System.out.print(cur.data+" ");
                if(cur.left!=null)
                    q.add(cur.left);
                if(cur.right!=null)
                    q.add(cur.right);
            }
        }
    }
    public static int countNodes(Node node){
        if(node==null)
            return 0;
        int ln = countNodes(node.left);
        int rn = countNodes(node.right);
        return ln+rn+1;
```

```
      }
}
class Test
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            obj.root.left.left = new Node(4);
            obj.root.left.right = new Node(5);
            obj.root.right.left = new Node(6);
            obj.root.right.right = new Node(7);
            System.out.println(BT.countNodes(obj.root));
      }
}
```

sum of nodes

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      public static void levelOrderTraversal(Node node){
            if(node==null)
                  return;
            Queue<Node> q = new LinkedList<>();
            q.add(node);
            q.add(null);
            while(!q.isEmpty()){
                  Node cur = q.remove();
                  if(cur==null){
                        System.out.println();
                        if(q.isEmpty())
```

```
                    break;
                else
                    q.add(null);
            }
            else{
                System.out.print(cur.data+" ");
                if(cur.left!=null)
                    q.add(cur.left);
                if(cur.right!=null)
                    q.add(cur.right);
            }
        }
    }
    public static int sumOfNodes(Node node){
        if(node==null)
            return 0;
        int ls = sumOfNodes(node.left);
        int rs = sumOfNodes(node.right);
        return ls+rs+node.data;
    }
}
class Test
{
    public static void main(String[] args)
    {
        BT obj = new BT();
        obj.root = new Node(7);
        obj.root.left = new Node(1);
        obj.root.right = new Node(2);
        obj.root.left.left = new Node(3);
        obj.root.left.right = new Node(4);
        obj.root.left.left.left = new Node(5);
        System.out.println(BT.sumOfNodes(obj.root));
    }
}

C:\test>javac Test.java

C:\test>java Test
22

C:\test>

height of tree
--------------
import java.util.*;
class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
class BT{
    Node root;
    BT(){
        root = null;
    }
    void inOrder(Node node){
        if(node==null)
            return;
        inOrder(node.left);
        System.out.print(node.data+" ");
        inOrder(node.right);
    }
    public static void levelOrderTraversal(Node node){
        if(node==null)
            return;
        Queue<Node> q = new LinkedList<>();
        q.add(node);
        q.add(null);
        while(!q.isEmpty()){
            Node cur = q.remove();
            if(cur==null){
                System.out.println();
                if(q.isEmpty())
                    break;
                else
                    q.add(null);
            }
            else{
                System.out.print(cur.data+" ");
                if(cur.left!=null)
                    q.add(cur.left);
                if(cur.right!=null)
                    q.add(cur.right);
            }
        }
    }
    public static int sumOfNodes(Node node){
        if(node==null)
            return 0;
        int ls = sumOfNodes(node.left);
        int rs = sumOfNodes(node.right);
        return ls+rs+node.data;
    }
    public static int height(Node node){
        if(node==null)
            return 0;
        int lh = height(node.left);
        int rh = height(node.right);
        return Math.max(lh,rh)+1;
    }
}
class Test
```

```
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            obj.root.left.left = new Node(4);
            obj.root.left.right = new Node(5);
            System.out.println(BT.height(obj.root));
      }
}

C:\test>javac Test.java

C:\test>java Test
3
```

search

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      public static void levelOrderTraversal(Node node){
            if(node==null)
                  return;
            Queue<Node> q = new LinkedList<>();
            q.add(node);
            q.add(null);
            while(!q.isEmpty()){
                  Node cur = q.remove();
                  if(cur==null){
                        System.out.println();
                        if(q.isEmpty())
```

```
                              break;
                        else
                              q.add(null);
                  }
                  else{
                        System.out.print(cur.data+" ");
                        if(cur.left!=null)
                              q.add(cur.left);
                        if(cur.right!=null)
                              q.add(cur.right);
                  }
            }
      }
      public static int sumOfNodes(Node node){
            if(node==null)
                  return 0;
            int ls = sumOfNodes(node.left);
            int rs = sumOfNodes(node.right);
            return ls+rs+node.data;
      }
      public static int height(Node node){
            if(node==null)
                  return 0;
            int lh = height(node.left);
            int rh = height(node.right);
            return Math.max(lh,rh)+1;
      }
      public static boolean search(Node node,int data){
            if(node==null)
                  return false;
            if(node.data == data)
                  return true;
            if(search(node.left,data))
                  return true;
            if(search(node.right,data))
                  return true;
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            obj.root.left.left = new Node(4);
            obj.root.left.right = new Node(5);
            System.out.println(BT.search(obj.root,2));//true
            System.out.println(BT.search(obj.root,6));//false
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
true
false

max element
-----------
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      public static void levelOrderTraversal(Node node){
            if(node==null)
                  return;
            Queue<Node> q = new LinkedList<>();
            q.add(node);
            q.add(null);
            while(!q.isEmpty()){
                  Node cur = q.remove();
                  if(cur==null){
                        System.out.println();
                        if(q.isEmpty())
                              break;
                        else
                              q.add(null);
                  }
                  else{
                        System.out.print(cur.data+" ");
                        if(cur.left!=null)
                              q.add(cur.left);
                        if(cur.right!=null)
                              q.add(cur.right);
                  }
            }
      }
      public static int maxElement(Node node){
            int max,left,right;
            if(node==null)
                  return Integer.MIN_VALUE;
            max = node.data;
            left = maxElement(node.left);
            right = maxElement(node.right);
            if(left>max)
                  max=left;
            if(right>max)
                  max=right;
            return max;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            obj.root.left.left = new Node(4);
            obj.root.left.right = new Node(5);
            System.out.println(BT.maxElement(obj.root));//5
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
5
```

min element

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      public static void levelOrderTraversal(Node node){
            if(node==null)
                  return;
            Queue<Node> q = new LinkedList<>();
            q.add(node);
            q.add(null);
            while(!q.isEmpty()){
                  Node cur = q.remove();
                  if(cur==null){
                        System.out.println();
                        if(q.isEmpty())
                              break;
                        else
                              q.add(null);
                  }
                  else{
                        System.out.print(cur.data+" ");
                        if(cur.left!=null)
                              q.add(cur.left);
                        if(cur.right!=null)
                              q.add(cur.right);
                  }
            }
      }
      public static int minElement(Node node){
            int min,left,right;
            if(node==null)
                  return Integer.MAX_VALUE;
            min = node.data;
            left = minElement(node.left);
            right = minElement(node.right);
            if(left<min)
                  min=left;
            if(right<min)
                  min=right;
            return min;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BT obj = new BT();
            obj.root = new Node(1);
            obj.root.left = new Node(2);
            obj.root.right = new Node(3);
            obj.root.left.left = new Node(4);
            obj.root.left.right = new Node(5);
            System.out.println(BT.minElement(obj.root));//1
      }
}

C:\test>javac Test.java

C:\test>java Test
1

equality
--------
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BT{
      Node root;
      BT(){
            root = null;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      public static void levelOrderTraversal(Node node){
            if(node==null)
                  return;
            Queue<Node> q = new LinkedList<>();
            q.add(node);
            q.add(null);
            while(!q.isEmpty()){
                  Node cur = q.remove();
                  if(cur==null){
                        System.out.println();
                        if(q.isEmpty())
                              break;
                        else
                              q.add(null);
                  }
                  else{
                        System.out.print(cur.data+" ");
                        if(cur.left!=null)
                              q.add(cur.left);
                        if(cur.right!=null)
```

```
                    q.add(cur.right);
            }
        }
    }
    public static boolean isEqual(Node node1,Node node2){
        if(node1==null && node2==null)
            return true;
        if(node1==null||node2==null)
            return false;
        else
            return
isEqual(node1.left,node2.left)&&isEqual(node1.right,node2.right)&&node1.d
ata==node2.data;
    }
}
class Test
{
    public static void main(String[] args)
    {
        BT obj1 = new BT();
        obj1.root = new Node(1);
        obj1.root.left = new Node(2);
        obj1.root.right = new Node(3);

        BT obj2 = new BT();
        obj2.root = new Node(1);
        obj2.root.left = new Node(2);
        obj2.root.right = new Node(3);

        System.out.println(BT.isEqual(obj1.root,obj2.root));
    }
}
```

```
C:\test>javac Test.java

C:\test>java Test
true
```

copy of tree

```
import java.util.*;
class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
class BT{
    Node root;
    BT(){
        root = null;
    }
    void inOrder(Node node){
        if(node==null)
              return;
        inOrder(node.left);
        System.out.print(node.data+" ");
        inOrder(node.right);
    }
    public static void levelOrderTraversal(Node node){
        if(node==null)
              return;
        Queue<Node> q = new LinkedList<>();
        q.add(node);
        q.add(null);
        while(!q.isEmpty()){
              Node cur = q.remove();
              if(cur==null){
                    System.out.println();
                    if(q.isEmpty())
                          break;
                    else
                          q.add(null);
              }
              else{
                    System.out.print(cur.data+" ");
                    if(cur.left!=null)
                          q.add(cur.left);
                    if(cur.right!=null)
                          q.add(cur.right);
              }
        }
    }
    public static Node copyTree(Node node){
        Node temp;
        if(node!=null){
              temp = new Node(node.data);
              temp.left = copyTree(node.left);
              temp.right = copyTree(node.right);
              return temp;
        }
        else
              return null;
    }
}
class Test
{
    public static void main(String[] args)
    {
        BT obj1 = new BT();
        obj1.root = new Node(1);
        obj1.root.left = new Node(2);
        obj1.root.right = new Node(3);
        BT obj2 = new BT();
        obj2.root = BT.copyTree(obj1.root);
        BT.levelOrderTraversal(obj1.root);
        BT.levelOrderTraversal(obj2.root);
    }
}
```

```
C:\test>javac Test.java

C:\test>java Test
1
2 3
1
2 3
```

Binary Search Tree (BST)
~~~~~~~~~~~~~~~~~~~~~~~~

introduction:

Binary search tree is special kind of binary tree with the following properties

1) elements which are existed in the left sub-tree of BST is less than root node.
2) elements which are existed in the right sub-tree of BST is greater than root node.
3) no duplicate keys is allowed.

creation of binary search tree by using inorder array

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      Node root;
      BST(){
            root = null;
      }
      static Node createBST(int[] nodes,int start,int end){
            Node node = null;
            if(start>end)
                  return null;
            int mid = (start+end)/2;
            node = new Node(nodes[mid]);
            node.left = createBST(nodes,start,mid-1);
```
~~~~~~~~~~~~~~~~~~~~~~~~

```
            node.right = createBST(nodes,mid+1,end);
            return node;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            int[] nodes = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
            obj.root = BST.createBST(nodes,0,nodes.length-1);
            System.out.println(obj.root.right.data);//5
      }
}
```

Insertion of a node into BST

Ex1:

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      Node root;
      BST(){
            root = null;
      }
      void insertNode(int data){
            root = insertNode(root,data);
      }
      Node insertNode(Node node,int data){
            if(node==null)
                  node = new Node(data);
            else{
                  if(data<node.data)
                        node.left = insertNode(node.left,data);
                  else
                        node.right = insertNode(node.right,data);
            }
            return node;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.insertNode(2);
            obj.insertNode(1);
            obj.insertNode(3);
            obj.insertNode(4);
            obj.inOrder(obj.root);//1-2-3-4
      }
}
```

Ex2:

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      Node root;
      BST(){
            root = null;
      }
      void insertNode(int data){
            root = insertNode(root,data);
      }
      Node insertNode(Node node,int data){
            if(node==null)
                  node = new Node(data);
            else{
                  if(data<node.data)
                        node.left = insertNode(node.left,data);
                  else
                        node.right = insertNode(node.right,data);
            }
            return node;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
```

```
    }
}
class Test
{
    public static void main(String[] args)
    {
        BST obj = new BST();
        obj.insertNode(6);
        obj.insertNode(4);
        obj.insertNode(2);
        obj.insertNode(5);
        obj.insertNode(1);
        obj.insertNode(3);
        obj.insertNode(8);
        obj.insertNode(7);
        obj.insertNode(9);
        obj.insertNode(10);
        obj.inOrder(obj.root);//1-2-3-4-5-6-7-8-9-10
    }
}
```

search operation in BST

```
import java.util.*;
class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
class BST{
    static Node root;
    BST(){
        root = null;
    }
    void insertNode(int data){
        root = insertNode(root,data);
    }
    Node insertNode(Node node,int data){
        if(node==null)
            node = new Node(data);
        else{
            if(data<node.data)
                node.left = insertNode(node.left,data);
            else
                node.right = insertNode(node.right,data);
        }
        return node;
    }
    void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      static boolean search(int value){
            Node curr = root;
            while(curr!=null){
                  if(value == curr.data)
                        return true;
                  else if(value<curr.data)
                        curr = curr.left;
                  else
                        curr = curr.right;
            }
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.insertNode(6);
            obj.insertNode(4);
            obj.insertNode(2);
            obj.insertNode(5);
            obj.insertNode(1);
            obj.insertNode(3);
            obj.insertNode(8);
            obj.insertNode(7);
            obj.insertNode(9);
            obj.insertNode(10);
            obj.inOrder(obj.root);//1-2-3-4-5-6-7-8-9-10
            System.out.println();
            System.out.println(BST.search(9));//true
            System.out.println(BST.search(11));//false
      }
}

C:\test>javac Test.java

C:\test>java Test
1 2 3 4 5 6 7 8 9 10
true
false

finding max and min element in BST
----------------------------------
import java.util.*;
class Node{
      int data;
      Node left;
```

```
    Node right;
    Node(int data){
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
class BST{
    static Node root;
    BST(){
        root = null;
    }
    void insertNode(int data){
        root = insertNode(root,data);
    }
    Node insertNode(Node node,int data){
        if(node==null)
            node = new Node(data);
        else{
            if(data<node.data)
                node.left = insertNode(node.left,data);
            else
                node.right = insertNode(node.right,data);
        }
        return node;
    }
    void inOrder(Node node){
        if(node==null)
            return;
        inOrder(node.left);
        System.out.print(node.data+" ");
        inOrder(node.right);
    }
    static boolean search(int value){
        Node curr = root;
        while(curr!=null){
            if(value == curr.data)
                return true;
            else if(value<curr.data)
                curr = curr.left;
            else
                curr = curr.right;
        }
        return false;
    }
    static Node findMaxNode(Node node){
        if(node==null)
            return null;
        while(node.right!=null)
            node = node.right;
        return node;
    }
    static Node findMinNode(Node node){
        if(node==null)
```

```
                  return null;
            while(node.left!=null)
                  node = node.left;
            return node;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.root = new Node(5);
            obj.root.left = new Node(3);
            obj.root.left.left = new Node(1);
            obj.root.left.right = new Node(4);
            obj.root.right = new Node(7);
            obj.root.right.left = new Node(6);
            obj.root.right.right = new Node(9);
            obj.inOrder(obj.root);//1-3-4-5-6-7-9
            System.out.println();
            System.out.println(BST.findMaxNode(obj.root).data);//9
            System.out.println(BST.findMinNode(obj.root).data);//9
      }
}

C:\test>javac Test.java

C:\test>java Test
1 3 4 5 6 7 9
9
1

The given tree is BST or not?
-----------------------------
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      static Node root;
      BST(){
            root = null;
      }
      void insertNode(int data){
            root = insertNode(root,data);
      }
      Node insertNode(Node node,int data){
            if(node==null)
                  node = new Node(data);
            else{
                  if(data<node.data)
                        node.left = insertNode(node.left,data);
                  else
                        node.right = insertNode(node.right,data);
            }
            return node;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      static boolean search(int value){
            Node curr = root;
            while(curr!=null){
                  if(value == curr.data)
                        return true;
                  else if(value<curr.data)
                        curr = curr.left;
                  else
                        curr = curr.right;
            }
            return false;
      }
      static Node findMaxNode(Node node){
            if(node==null)
                  return null;
            while(node.right!=null)
                  node = node.right;
            return node;
      }
      static Node findMinNode(Node node){
            if(node==null)
                  return null;
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static boolean isBST(Node node){
            if(node==null)
                  return true;
            if(node.left!=null && findMaxNode(node.left).data>node.data)
                  return false;
            if(node.right!=null &&
findMinNode(node.right).data<node.data)
                  return false;
            return isBST(node.left) && isBST(node.right);
      }
}
```

```
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.root = new Node(3);
            obj.root.left = new Node(2);
            obj.root.left.left = new Node(6);
            obj.root.right = new Node(5);
            obj.root.right.left = new Node(4);
            obj.root.right.right = new Node(9);
            obj.inOrder(obj.root);//1-2-3-4-5-9
            System.out.println();
            System.out.println(BST.isBST(obj.root));//true
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
6 2 3 4 5 9
false
```

```
delete operation:
-----------------
case1: deleting a node which is not having any child nodes
case2: deleting a node which is having single child
case3: deleting a node which is having two children
```

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      static Node root;
      BST(){
            root = null;
      }
      void insertNode(int data){
            root = insertNode(root,data);
      }
      Node insertNode(Node node,int data){
            if(node==null)
                  node = new Node(data);
            else{
                  if(data<node.data)
```

```
                        node.left = insertNode(node.left,data);
                  else
                        node.right = insertNode(node.right,data);
            }
            return node;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      static boolean search(int value){
            Node curr = root;
            while(curr!=null){
                  if(value == curr.data)
                        return true;
                  else if(value<curr.data)
                        curr = curr.left;
                  else
                        curr = curr.right;
            }
            return false;
      }
      static Node findMaxNode(Node node){
            if(node==null)
                  return null;
            while(node.right!=null)
                  node = node.right;
            return node;
      }
      static Node findMinNode(Node node){
            if(node==null)
                  return null;
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static Node delete(Node node,int value){
            if(node.data <value)
                  node.right = delete(node.right,value);
            else if(node.data>value)
                  node.left = delete(node.left,value);
            else{
                  //case1: no child (leaf)
                  if(node.left==null && node.right==null)
                        return null;
                  //case2: one child
                  else if(node.left==null)
                        return node.right;
                  else if(node.right==null)
                        return node.left;
                  //case3: two children
                  Node is = findInOrderSuccessor(node.right);
                  node.data = is.data;
                  node.right = delete(node.right,is.data);
            }
            return node;
      }
      static Node findInOrderSuccessor(Node node){
            while(node.left!=null)
                  node = node.left;
            return node;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.root = new Node(10);
            obj.root.left = new Node(5);
            obj.root.right = new Node(15);
            obj.root.left.left = new Node(3);
            obj.root.left.right = new Node(7);
            obj.root.left.right.left = new Node(6);
            obj.root.right.left = new Node(12);
            obj.root.right.right = new Node(17);
            obj.root.right.right.left = new Node(16);
            obj.root.right.right.right = new Node(18);
            obj.inOrder(obj.root);//3-5-6-7-10-12-15-16-17-18
            System.out.println();
            BST.delete(obj.root,17);
            obj.inOrder(obj.root);//5-6-7-10-12-15-16-17-18
      }
}
```

```
C:\test>java Test
3 5 6 7 10 12 15 16 17 18
3 5 6 7 10 12 15 16 18
```

Remove all leaf nodes from BST

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      static Node root;
      BST(){
```

```
      root = null;
}
void insertNode(int data){
      root = insertNode(root,data);
}
Node insertNode(Node node,int data){
      if(node==null)
            node = new Node(data);
      else{
            if(data<node.data)
                  node.left = insertNode(node.left,data);
            else
                  node.right = insertNode(node.right,data);
      }
      return node;
}
void inOrder(Node node){
      if(node==null)
            return;
      inOrder(node.left);
      System.out.print(node.data+" ");
      inOrder(node.right);
}
static boolean search(int value){
      Node curr = root;
      while(curr!=null){
            if(value == curr.data)
                  return true;
            else if(value<curr.data)
                  curr = curr.left;
            else
                  curr = curr.right;
      }
      return false;
}
static Node findMaxNode(Node node){
      if(node==null)
            return null;
      while(node.right!=null)
            node = node.right;
      return node;
}
static Node findMinNode(Node node){
      if(node==null)
            return null;
      while(node.left!=null)
            node = node.left;
      return node;
}
static Node delete(Node node,int value){
      if(node.data <value)
            node.right = delete(node.right,value);
      else if(node.data>value)
            node.left = delete(node.left,value);
            else{
                  //case1: no child (leaf)
                  if(node.left==null && node.right==null)
                        return null;
                  //case2: one child
                  else if(node.left==null)
                        return node.right;
                  else if(node.right==null)
                        return node.left;
                  //case3: two children
                  Node is = findInOrderSuccessor(node.right);
                  node.data = is.data;
                  node.right = delete(node.right,is.data);
            }
            return node;
      }
      static Node findInOrderSuccessor(Node node){
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static Node leafDelete(Node node){
            if(node==null)
                  return null;
            if(node.left==null && node.right==null)
                  return null;
            node.left=leafDelete(node.left);
            node.right=leafDelete(node.right);
            return node;
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.root = new Node(4);
            obj.root.left = new Node(2);
            obj.root.right = new Node(6);
            obj.root.left.left = new Node(1);
            obj.root.left.right = new Node(3);
            obj.root.right.left = new Node(5);
            obj.root.right.right = new Node(7);
            obj.inOrder(obj.root);//1-2-3-4-5-6-7
            System.out.println();
            BST.leafDelete(obj.root);
            obj.inOrder(obj.root);//2-4-6
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
```

```
1 2 3 4 5 6 7
2 4 6
```

Print in range
----

print list of nodes whose values are in between k1 and k2

```
import java.util.*;
class Node{
      int data;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
      }
}
class BST{
      static Node root;
      BST(){
            root = null;
      }
      void insertNode(int data){
            root = insertNode(root,data);
      }
      Node insertNode(Node node,int data){
            if(node==null)
                  node = new Node(data);
            else{
                  if(data<node.data)
                        node.left = insertNode(node.left,data);
                  else
                        node.right = insertNode(node.right,data);
            }
            return node;
      }
      void inOrder(Node node){
            if(node==null)
                  return;
            inOrder(node.left);
            System.out.print(node.data+" ");
            inOrder(node.right);
      }
      static boolean search(int value){
            Node curr = root;
            while(curr!=null){
                  if(value == curr.data)
                        return true;
                  else if(value<curr.data)
                        curr = curr.left;
                  else
                        curr = curr.right;
            }
            return false;
      }
      static Node findMaxNode(Node node){
            if(node==null)
                  return null;
            while(node.right!=null)
                  node = node.right;
            return node;
      }
      static Node findMinNode(Node node){
            if(node==null)
                  return null;
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static Node delete(Node node,int value){
            if(node.data <value)
                  node.right = delete(node.right,value);
            else if(node.data>value)
                  node.left = delete(node.left,value);
            else{
                  //case1: no child (leaf)
                  if(node.left==null && node.right==null)
                        return null;
                  //case2: one child
                  else if(node.left==null)
                        return node.right;
                  else if(node.right==null)
                        return node.left;
                  //case3: two children
                  Node is = findInOrderSuccessor(node.right);
                  node.data = is.data;
                  node.right = delete(node.right,is.data);
            }
            return node;
      }
      static Node findInOrderSuccessor(Node node){
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static void printInRange(Node node,int k1,int k2){
            if(node==null)
                  return;
            if(node.data >= k1 && node.data <=k2){
                  printInRange(node.left,k1,k2);
                  System.out.print(node.data+" ");
                  printInRange(node.right,k1,k2);
            }
            else if(node.data < k1){
                  printInRange(node.left,k1,k2);
            }
            else{
```

```
                        printInRange(node.right,k1,k2);
                }
        }
}
class Test
{
        public static void main(String[] args)
        {
                BST obj = new BST();
                obj.root = new Node(8);
                obj.root.left = new Node(5);
                obj.root.right = new Node(10);
                obj.root.left.left = new Node(3);
                obj.root.left.right = new Node(6);
                obj.root.left.left.left = new Node(1);
                obj.root.left.left.right = new Node(4);
                obj.root.right.right = new Node(11);
                obj.root.right.right.right = new Node(14);
                obj.inOrder(obj.root);//1 3 4 5 6 8 10 11 14
                System.out.println();
                obj.printInRange(obj.root,5,12);//5 6 8 10 11
        }
}
```

```
C:\test>java Test
1 3 4 5 6 8 10 11 14
5 6 8 10 11
```

Root to leaf path

```
import java.util.*;
class Node{
        int data;
        Node left;
        Node right;
        Node(int data){
                this.data = data;
                this.left = null;
                this.right = null;
        }
}
class BST{
        static Node root;
        BST(){
                root = null;
        }
        void insertNode(int data){
                root = insertNode(root,data);
        }
        Node insertNode(Node node,int data){
                if(node==null)
                        node = new Node(data);
                else{
```

```
            if(data<node.data)
                node.left = insertNode(node.left,data);
            else
                node.right = insertNode(node.right,data);
        }
        return node;
    }
    void inOrder(Node node){
        if(node==null)
            return;
        inOrder(node.left);
        System.out.print(node.data+" ");
        inOrder(node.right);
    }
    static boolean search(int value){
        Node curr = root;
        while(curr!=null){
            if(value == curr.data)
                return true;
            else if(value<curr.data)
                curr = curr.left;
            else
                curr = curr.right;
        }
        return false;
    }
    static Node findMaxNode(Node node){
        if(node==null)
            return null;
        while(node.right!=null)
            node = node.right;
        return node;
    }
    static Node findMinNode(Node node){
        if(node==null)
            return null;
        while(node.left!=null)
            node = node.left;
        return node;
    }
    static Node delete(Node node,int value){
        if(node.data <value)
            node.right = delete(node.right,value);
        else if(node.data>value)
            node.left = delete(node.left,value);
        else{
            //case1: no child (leaf)
            if(node.left==null && node.right==null)
                return null;
            //case2: one child
            else if(node.left==null)
                return node.right;
            else if(node.right==null)
                return node.left;
```

```
                  //case3: two children
                  Node is = findInOrderSuccessor(node.right);
                  node.data = is.data;
                  node.right = delete(node.right,is.data);
            }
            return node;
      }
      static Node findInOrderSuccessor(Node node){
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static void printRootToLeaf(Node node,ArrayList<Integer> path){
            if(node==null)
                  return;
            path.add(node.data);
            if(node.left==null && node.right==null)
                  printPath(path);
            printRootToLeaf(node.left,path);
            printRootToLeaf(node.right,path);
            path.remove(path.size()-1);
      }
      static void printPath(ArrayList<Integer> path){
            for(int i=0;i<path.size();i++)
                  System.out.print(path.get(i)+" -> ");
            System.out.println("null");
      }
}
class Test
{
      public static void main(String[] args)
      {
            BST obj = new BST();
            obj.root = new Node(8);
            obj.root.left = new Node(5);
            obj.root.right = new Node(10);
            obj.root.left.left = new Node(3);
            obj.root.left.right = new Node(6);
            obj.root.left.left.left = new Node(1);
            obj.root.left.left.right = new Node(4);
            obj.root.right.right = new Node(11);
            obj.root.right.right.right = new Node(14);
            obj.inOrder(obj.root);//1 3 4 5 6 8 10 11 14
            System.out.println();
            obj.printRootToLeaf(obj.root,new ArrayList<Integer>());
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
1 3 4 5 6 8 10 11 14
8 -> 5 -> 3 -> 1 -> null
```

8 -> 5 -> 3 -> 4 -> null
8 -> 5 -> 6 -> null
8 -> 10 -> 11 -> 14 -> null

Tree data structure
Binary Tree data structure
Binary Search Tree data structure

Height Balanced Trees ------> AVL, RedBlack, B, B+ etc

AVL Trees

=> AVL trees are self-balancing trees.
=> Balance factor (bf) of an AVL tree is always -1, 0, +1.
=> Balance Factor (BF) = height(left sub-tree) - height(right sub-tree)
=> |HL - HR| < 2
=> BF = 0 or BF = -1 or BF = +1 , then it is balanced tree (AVL Tree).

Ex:

Insert --> 10,20,30
number of nodes = 3
no of BST's = 3! = 6

AVL Trees:

LL case---> Right Rotation
LR case --> Left Rotation, Right Rotation
RR case --> Left Rotation
RL case --> Right Rotation, Left Rotation

Left Rotation and Right Rotation

creation of AVL Trees

Insert ---> 40, 20, 10, 25, 30, 22, 50

https://cmps-people.ok.ubc.ca/ylucet/DS/AVLtree.html

Insert, Delete and Search ---> Same linke BST

Implementation of AVL Tree

```
import java.util.*;
class Node{
      int data;
      int ht;
      Node left;
      Node right;
      Node(int data){
            this.data = data;
            this.left = null;
            this.right = null;
            this.ht = 1;
      }
}
class AVLTree{
      static Node root;
      AVLTree(){
            root = null;
      }
      static int height(Node node){
            if(node==null)
                  return 0;
            return node.ht;
      }
      static int getBalance(Node node){//balance factor HL-HR
            if(node==null)
                  return 0;
            return height(node.left)-height(node.right);
      }
      static void insert(int value){
            root = insert(root,value);
      }
      static Node insert(Node node,int value){
            if(node==null)
                  return new Node(value);
            if(value < node.data)
                  node.left = insert(node.left,value);
            else if(value > node.data)
                  node.right = insert(node.right,value);
            else
                  return node;//duplicate nodes
            //balancing
            node.ht = 1 + Math.max(height(node.left),height(node.right));
            int bf = getBalance(node);
            if(bf>1 && value < node.left.data){//LL case
                  return rightRotation(node);
            }
            if(bf<-1 && value > node.right.data){//RR case
                  return leftRotation(node);
            }
            if(bf>1 && value > node.left.data){//LR case
                  node.left = leftRotation(node.left);
                  return rightRotation(node);
            }
            if(bf<-1 && value<node.right.data){//RL case
                  node.right = rightRotation(node.right);
                  return leftRotation(node);
            }
            return node;
      }
      static void delete(int value){
            root = delete(root,value);
      }
      static Node delete(Node node,int value){
            if(node.data < value)
                  node.right = delete(node.right,value);
            else if(node.data > value)
                  node.left = delete(node.left,value);
            else{
                  if(node.left==null && node.right==null)//case1:no
children
                        return null;
                  if(node.left==null)//case2: one child (right)
                        return node.right;
                  else if(node.right==null) //case2: one child (left)
                        return node.left;
                  Node is = findInOrderSuccessor(node.right);
                  node.data = is.data;
                  node.right = delete(node.right,is.data);
            }
            if(node==null)
                  return node;
            node.ht = Math.max(height(node.left),height(node.right))+1;
            int bf = getBalance(node);
            if(bf>1 && getBalance(node.left)>=0)
                  return rightRotate(node);
            if(bf>1 && getBalance(node.left)<0)
            {
                  node.left = leftRotate(node.left);
                  return rightRotate(node);
            }
            if(bf<-1 && getBalance(node.right)<=0)
                  return leftRotate(node);
            if(bf<-1 && getBalance(node.right)>0)
            {
                  node.right = rightRotate(node.right);
                  return leftRotate(node);
            }
            return node;
      }
      static Node findInOrderSuccessor(Node node){
            while(node.left!=null)
                  node = node.left;
            return node;
      }
      static void preOrder(Node node){
            if(node==null)
                  return;
            System.out.print(node.data+" ");
            preOrder(node.left);
            preOrder(node.right);
      }
      static Node rightRotation(Node y){
            Node x = y.left;
            Node T = x.right;
            x.right = y;
            y.left = T;
            x.ht = 1+Math.max(height(x.left),height(x.right));
            y.ht = 1+Math.max(height(y.left),height(y.right));
            return x;
```

```
	}
	static Node leftRotation(Node x){
		Node y = x.right;
		Node T = y.left;
		y.left = x;
		x.right = T;
		x.ht = 1+Math.max(height(x.left),height(x.right));
		y.ht = 1+Math.max(height(y.left),height(y.right));
		return y;
	}
}
class Test
{
	public static void main(String[] args)
	{
		AVLTree obj = new AVLTree();
		AVLTree.insert(10);
		AVLTree.insert(20);
		AVLTree.insert(30);
		AVLTree.insert(40);
		AVLTree.insert(50);
		AVLTree.insert(25);
		AVLTree.preOrder(obj.root);
		//output: 30 20 10 25 40 50
		//BST: 10-20-30-25-40-50
	}
}
```

Backtracking:

backtracking a method by which a solution is found by exahaustively searching through a large volume but finate number of state with some boundary condition.

Ex:

Rat in maze
N-Queen
subsets
permutations
towers of hanoi
grid based problems etc

Types of backtracking problems:

```
1) Decision ---------------------> Yes / No
2) Optimized Solution -----------> One sol
3) Enumerations (all sols) -----> all sols
```

Backtracking with Arrays

```
import java.util.*;
class Test
```

```
{
    public static void changeArray(int[] a,int index,int value){
        if(index==a.length){//base condition
            System.out.println(Arrays.toString(a));
            return;
        }
        a[index] = value;
        changeArray(a,index+1,value+1);//recursion
        a[index] = a[index]-2;  //backtracking
    }
    public static void main(String[] args)
    {
        int[] a = new int[5];
        System.out.println(Arrays.toString(a));
        changeArray(a,0,1);
        System.out.println(Arrays.toString(a));
    }
}
```

```
[0, 0, 0, 0, 0]
[1, 2, 3, 4, 5]
[-1, 0, 1, 2, 3]
```

Find Subsets:
----

find all subsets of the given string.

Ex: abc ----> "",a,b,c,ab,bc,ac,abc

3 char ----> 2^n ---> 2^3 = 8

abcd ----> 2^4 = 16

```
import java.util.*;
class Test
{
    public static void findSubsets(String s,String ans,int index){
        //base condition
        if(index==s.length()){
            if(ans.length()==0)
                System.out.println("null");
            else
                System.out.println(ans);
            return;
        }
        //yes condition
        findSubsets(s,ans+s.charAt(index),index+1);
        //no condition
        findSubsets(s,ans,index+1);
    }
    public static void main(String[] args)
    {
        String s = "abc";
        String ans = "";
        findSubsets(s,ans,0);
```

```
      }
}

C:\test>javac Test.java

C:\test>java Test
abc
ab
ac
a
bc
b
c
null
```

find permutations

"abc" ----> abc, acb, bac, bca, cab, cba

n! number of permutations

3! = 6

```
import java.util.*;
class Test
{
      static void findPermutations(String s,String ans){
            //base condition
            if(s.length()==0){
                  System.out.println(ans);
                  return;
            }
            //recursion
            for(int i=0;i<s.length();i++){
                  char cur = s.charAt(i);
                  String ns = s.substring(0,i)+s.substring(i+1);
                  findPermutations(ns,ans+cur);
            }
      }
      public static void main(String[] args)
      {
            String s = "abc";
            findPermutations(s,"");
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
abc
acb
bac
bca
cab
```

cba

N-queens problem:

We have to place N-queens on NxN chess board such that no 2 queens can attack each other

```
import java.util.*;
class Test
{
    static int counter;
    public static void printBoard(char[][] board){
        System.out.println("---chess board---");
        for(int i=0;i<board.length;i++){
            for(int j=0;j<board.length;j++){
                System.out.print(board[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static boolean isSafe(char[][] board,int row,int col){
        //vertical up
        for(int i=row-1;i>=0;i--){
            if(board[i][col]=='Q')
                return false;
        }
        //dia left up
        for(int i=row-1,j=col-1;i>=0 && j>=0;i--,j--){
            if(board[i][j]=='Q')
                return false;
        }
        //diag right up
        for(int i=row-1,j=col+1;i>=0 && j<board.length;i--,j++){
            if(board[i][j]=='Q')
                return false;
        }
        return true;
    }
    public static void nQueens(char[][] board,int row){
        //base condition
        if(row==board.length){
            printBoard(board);
            counter++;
            return;
        }
        //main logic (recursion and backtracking)
        for(int j=0;j<board.length;j++){//col loop
            if(isSafe(board,row,j)){
                board[row][j] = 'Q';
                nQueens(board,row+1);//recursion
                board[row][j] = 'x';
            }
        }
    }
```

```
      public static void main(String[] args)
      {
            int n = 5;
            char board[][] = new char[n][n];
            for(int i=0;i<board.length;i++){
                  for(int j=0;j<board.length;j++){
                        board[i][j]='x';
                  }
            }
            nQueens(board,0);
            System.out.println("Number of solutions: "+counter);
      }
}

C:\test>javac Test.java

C:\test>java Test
---chess board---
Q x x x x
x x Q x x
x x x x Q
x Q x x x
x x x Q x
---chess board---
Q x x x x
x x x Q x
x Q x x x
x x x x Q
x x Q x x
---chess board---
x Q x x x
x x x Q x
Q x x x x
x x Q x x
x x x x Q
---chess board---
x Q x x x
x x x x Q
x x Q x x
Q x x x x
x x x Q x
---chess board---
x x Q x x
Q x x x x
x x x Q x
x Q x x x
x x x x Q
---chess board---
x x Q x x
x x x x Q
x Q x x x
x x x Q x
Q x x x x
---chess board---
```

```
x x x Q x
Q x x x x
x x Q x x
x x x x Q
x Q x x x
---chess board---
x x x Q x
x Q x x x
x x x x Q
x x Q x x
Q x x x x
---chess board---
x x x x Q
x Q x x x
x x x Q x
Q x x x x
x x Q x x
---chess board---
x x x x Q
x x Q x x
Q x x x x
x x x Q x
x Q x x x
Number of solutions: 10
```

Grid ways:

Find number of ways to reach from (0,0) to (N-1,N-1) in a NxN grid matrix

allowed moves are ----> right or down

```
import java.util.*;
class Test
{
	public static int gridWays(int i,int j,int n,int m){
		//base
		if(i==n-1 && j==m-1)
			return 1;
		else if(i==n || j==m)
			return 0;
		int value1 = gridWays(i+1,j,n,m);
		int value2 = gridWays(i,j+1,n,m);
		return value1+value2;
	}
	public static void main(String[] args)
	{
		int n=4,m=4;
		System.out.println(gridWays(0,0,n,m));
	}
}
```

```
C:\test>javac Test.java

C:\test>java Test
```

```
6

C:\test>javac Test.java

C:\test>java Test
20

sudoku problem solver
---------------------

9x9

{
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0},
      {0,0,0,0,0,0,0,0,0}
}

problem and solution

import java.util.*;
class Test
{
      public static boolean isSafe(int sudoku[][],int row,int col,int
digit){
            //column
            for(int i=0;i<=8;i++){
                  if(sudoku[i][col]==digit)
                        return false;
            }
            //row
            for(int j=0;j<=8;j++){
                  if(sudoku[row][j]==digit)
                        return false;
            }
            //grid
            int sr = (row/3)*3;
            int sc = (col/3)*3;
            //3x3 grid
            for(int i=sr;i<sr+3;i++){
                  for(int j=sc;j<sc+3;j++){
                        if(sudoku[i][j]==digit)
                              return false;
                  }
            }
            return true;
      }
      public static boolean sudokuSolver(int sudoku[][],int row,int col){
        //base
        if(row==9 && col==0)
            return true;
        //recursion
        int nextRow = row, nextCol = col+1;
        if(col+1==9){
            nextRow = row+1;
            nextCol = 0;
        }
        if(sudoku[row][col]!=0)
            return sudokuSolver(sudoku,nextRow,nextCol);
        for(int digit=1;digit<=9;digit++){
            if(isSafe(sudoku,row,col,digit)){
                sudoku[row][col] = digit;
                if(sudokuSolver(sudoku,nextRow,nextCol)){
                    return true;
                }
                sudoku[row][col] = 0;
            }
        }
        return false;
    }
    public static void printSudoku(int sudoku[][]){
        for(int i=0;i<9;i++){
            for(int j=0;j<9;j++){
                System.out.print(sudoku[i][j]+" ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args)
    {
        int sudoku[][]={
            {0,0,8,0,0,0,0,0,0},
            {4,9,0,1,5,7,0,0,2},
            {0,0,3,0,0,4,1,9,0},
            {1,8,5,0,6,0,0,2,0},
            {0,0,0,0,2,0,0,6,0},
            {9,6,0,4,0,5,3,0,0},
            {0,3,0,0,7,2,0,0,4},
            {0,4,9,0,3,0,0,5,7},
            {8,2,7,0,0,9,0,1,3}
        };
        printSudoku(sudoku);
        if(sudokuSolver(sudoku,0,0)){
            System.out.println("solution exists");
            printSudoku(sudoku);
        }
        else
            System.out.println("solution not exists");
    }
}
```

```
C:\test>javac Test.java

C:\test>java Test
0 0 8 0 0 0 0 0 0
4 9 0 1 5 7 0 0 2
0 0 3 0 0 4 1 9 0
1 8 5 0 6 0 0 2 0
0 0 0 0 2 0 0 6 0
9 6 0 4 0 5 3 0 0
0 3 0 0 7 2 0 0 4
0 4 9 0 3 0 0 5 7
8 2 7 0 0 9 0 1 3
solution exists
2 1 8 3 9 6 7 4 5
4 9 6 1 5 7 8 3 2
7 5 3 2 8 4 1 9 6
1 8 5 7 6 3 4 2 9
3 7 4 9 2 8 5 6 1
9 6 2 4 1 5 3 7 8
5 3 1 6 7 2 9 8 4
6 4 9 8 3 1 2 5 7
8 2 7 5 4 9 6 1 3
```

another:

```
C:\test>javac Test.java

C:\test>java Test
5 4 0 0 2 0 8 0 6
0 1 9 0 0 7 0 0 3
0 0 0 3 0 0 2 1 0
9 0 0 4 0 5 0 2 0
0 0 1 0 0 0 6 0 4
6 0 4 0 3 2 0 8 0
0 6 0 0 0 0 1 9 0
4 0 2 0 0 9 0 0 5
0 9 0 0 7 0 4 0 2
solution exists
5 4 3 9 2 1 8 7 6
2 1 9 6 8 7 5 4 3
8 7 6 3 5 4 2 1 9
9 8 7 4 6 5 3 2 1
3 2 1 7 9 8 6 5 4
6 5 4 1 3 2 9 8 7
7 6 5 2 4 3 1 9 8
4 3 2 8 1 9 7 6 5
1 9 8 5 7 6 4 3 2
```

Graphs Data Structure:
~~~~~~~~~~~~~~~~~~~~~~

introduction:

~~~~~~~~~~~~~~~~~~~~~~

```
==> collection of nodes
==> nodes or vertex or vertices
==> network of nodes
==> connections --> edges

edge
node or vertex
directed graph
undirected graph
weighted graph
unweighted graph


Representation of graph:
~~~~~~~~~~~~~~~~~~~~~~~~
1) Adjacency list (list of lists)
2) Adjacency matrix (2-d array)

import java.util.*;
class Edge{
      int s,d,w;
      Edge(int s,int d,int w){
            this.s = s;
            this.d = d;
            this.w = w;
      }
      public String toString(){
            return "{"+this.s+","+this.d+","+this.w+"}";
      }
}
class Test
{
      public static void main(String[] args)
      {
            int v = 5;
            ArrayList<Edge>[] graph = new ArrayList[v];
            for(int i =0;i<v;i++)
                  graph[i] = new ArrayList<Edge>();

            //0th vertex
            graph[0].add(new Edge(0,1,5));

            //1st vertex
            graph[1].add(new Edge(1,0,5));
            graph[1].add(new Edge(1,2,1));
            graph[1].add(new Edge(1,3,3));

            //2nd vertex
            graph[2].add(new Edge(2,1,1));
            graph[2].add(new Edge(2,3,1));
            graph[2].add(new Edge(2,4,2));

            //3rd vertex
            graph[3].add(new Edge(3,1,3));
```

```
        graph[3].add(new Edge(3,2,1));

        //4th vertex
        graph[4].add(new Edge(4,2,2));

        /*
        //all dest verteces for node 2
        for(int i=0;i<graph[2].size();i++){
            Edge e = graph[2].get(i);
            System.out.println(e.d);
        }*/

        for(ArrayList L:graph){
            System.out.println(L);
        }
    }
}
```

```
C:\test>java Test
[{0,1,5}]
[{1,0,5}, {1,2,1}, {1,3,3}]
[{2,1,1}, {2,3,1}, {2,4,2}]
[{3,1,3}, {3,2,1}]
[{4,2,2}]
```

```
Adjacency Matrix Representation of a graph
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Graph{
    int v;
    int[][] adj;
    Graph(int v){
        this.v = v;
        adj = new int[v][v];
    }
    void addDirectedEdge(int src,int dest,int cost){
        adj[src][dest] = cost;
    }
    void addUnDirectedEdge(int src,int dest,int cost){
        addDirectedEdge(src,dest,cost);
        addDirectedEdge(dest,src,cost);
    }
    void printGraph(){
        for(int i=0;i<v;i++){
            System.out.print("vertex "+i+" is connected to: ");
            for(int j=0;j<v;j++){
                if(adj[i][j]!=0)
                    System.out.print("("+j+", "+adj[i][j]+") ");
            }
```

```
                  System.out.println();
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            //from vertex 0
            g.addUnDirectedEdge(0,1,1);
            g.addUnDirectedEdge(0,2,1);
            //from vertext 1
            g.addUnDirectedEdge(1,2,1);
            g.addUnDirectedEdge(1,3,1);
            //from vertext 2
            g.addUnDirectedEdge(2,3,1);

            g.printGraph();
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to: (1, 1) (2, 1)
vertex 1 is connected to: (0, 1) (2, 1) (3, 1)
vertex 2 is connected to: (0, 1) (1, 1) (3, 1)
vertex 3 is connected to: (1, 1) (2, 1)
```

```
Adjacency List Representation:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
```

```
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            //from vertex 0
            g.addUnDirectedEdge(0,1,1);
            g.addUnDirectedEdge(0,2,1);
            //from vertext 1
            g.addUnDirectedEdge(1,2,1);
            g.addUnDirectedEdge(1,3,1);
            //from vertext 2
            g.addUnDirectedEdge(2,3,1);

            g.printGraph();
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=> (0, 1) (2, 1) (3, 1)
vertex 2 is connected to=> (0, 1) (1, 1) (3, 1)
vertex 3 is connected to=> (1, 1) (2, 1)
```

Graph Traversal
----

visiting every vertex is called as graph traversal

DFS and BFS

```
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static void bfs(Graph g,int source){
            int v = g.v;
            boolean[] visited = new boolean[v];
            LinkedList<Integer> q = new LinkedList<Integer>();
            q.add(source);
            visited[source]=true;
            while(!q.isEmpty()){
                  int curr = q.remove();
            System.out.print(curr+" ");
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp){
                if(visited[e.dest]==false){
                    visited[e.dest] = true;
                    q.add(e.dest);
                }
            }
        }
    }
    static void dfs(Graph g,int curr,boolean[] visited){
        System.out.print(curr+" ");
        visited[curr] = true;
        LinkedList<Edge> temp = g.adj.get(curr);
        for(Edge e:temp){
            if(visited[e.dest]==false){
                dfs(g,e.dest,visited);
            }
        }
    }
}
class Test
{
    public static void main(String[] args)
    {
        Graph g = new Graph(5);
        g.addUnDirectedEdge(0,1);
        g.addUnDirectedEdge(1,2);
        g.addUnDirectedEdge(1,4);
        g.addUnDirectedEdge(2,3);
        g.addUnDirectedEdge(3,4);
        g.printGraph();
        System.out.print("BFS: ");
        Graph.bfs(g,0);//BFS: 0 1 2 4 3
        System.out.println();
        System.out.print("DFS: ");
        Graph.dfs(g,0,new boolean[g.v]);//DFS: 0 1 2 3 4
        /*
        Graph g = new Graph(4);
        g.addUnDirectedEdge(0,1);
        g.addUnDirectedEdge(0,2);
        g.addUnDirectedEdge(1,2);
        g.addUnDirectedEdge(2,3);
        g.printGraph();
        System.out.print("BFS: ");
        Graph.bfs(g,0);//BFS: 0 1 2 3
        System.out.println();
        System.out.print("DFS: ");
        Graph.dfs(g,0,new boolean[g.v]);//DFS: 0 1 2 3
        */
    }
}
```

determineing a path from vertex 'u' to vertex 'v'.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

for given source and destination, determine if a path is existed from source to dest or not.

```
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static void bfs(Graph g,int source){
            int v = g.v;
            boolean[] visited = new boolean[v];
            LinkedList<Integer> q = new LinkedList<Integer>();
            q.add(source);
            visited[source]=true;
```

```
            while(!q.isEmpty()){
                  int curr = q.remove();
                  System.out.print(curr+" ");
                  LinkedList<Edge> temp = g.adj.get(curr);
                  for(Edge e:temp){
                        if(visited[e.dest]==false){
                              visited[e.dest] = true;
                              q.add(e.dest);
                        }
                  }
            }
      }
      static void dfs(Graph g,int curr,boolean[] visited){
            System.out.print(curr+" ");
            visited[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp){
                  if(visited[e.dest]==false){
                        dfs(g,e.dest,visited);
                  }
            }
      }
      static boolean hasPath(Graph g,int source,int dest){
            boolean[] visited = new boolean[g.v];
            dfsUtil(g,source,visited);
            return visited[dest];
      }
      static void dfsUtil(Graph g,int curr,boolean visited[])
      {
            visited[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp)
            {
                  if(visited[e.dest]==false)
                        dfsUtil(g,e.dest,visited);
            }
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(6);
            g.addUnDirectedEdge(0,1);
            g.addUnDirectedEdge(0,2);
            g.addUnDirectedEdge(1,3);
            g.addUnDirectedEdge(1,4);
            g.addUnDirectedEdge(2,3);
            g.addUnDirectedEdge(3,4);
            g.printGraph();
            System.out.println(Graph.hasPath(g,0,4));//true
            System.out.println(Graph.hasPath(g,0,5));//false
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=> (0, 1) (3, 1) (4, 1)
vertex 2 is connected to=> (0, 1) (3, 1)
vertex 3 is connected to=> (1, 1) (2, 1) (4, 1)
vertex 4 is connected to=> (1, 1) (3, 1)
vertex 5 is connected to=>
true
false
```

Count all paths from source to destination

~~~~~~

Given source and destination verteces. Count of all paths from source to destination.

```
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
```
~~~~~~

```
        for(int i=0;i<v;i++){
            LinkedList<Edge> temp = adj.get(i);
            System.out.print("vertex "+i+" is connected to=> ");
            for(Edge e : temp){
                System.out.print("("+e.dest+", "+e.cost+") ");
            }
            System.out.println();
        }
    }
    static void bfs(Graph g,int source){
        int v = g.v;
        boolean[] visited = new boolean[v];
        LinkedList<Integer> q = new LinkedList<Integer>();
        q.add(source);
        visited[source]=true;
        while(!q.isEmpty()){
            int curr = q.remove();
            System.out.print(curr+" ");
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp){
                if(visited[e.dest]==false){
                    visited[e.dest] = true;
                    q.add(e.dest);
                }
            }
        }
    }
    static void dfs(Graph g,int curr,boolean[] visited){
        System.out.print(curr+" ");
        visited[curr] = true;
        LinkedList<Edge> temp = g.adj.get(curr);
        for(Edge e:temp){
            if(visited[e.dest]==false){
                dfs(g,e.dest,visited);
            }
        }
    }
    static int countAllPaths(Graph g,int source,int dest)
    {
        boolean[] visited = new boolean[g.v];
        return countAllPaths(g,visited,source,dest);
    }
    static int countAllPaths(Graph g,boolean[] visited,int source,int
dest)
    {
        if(source==dest)
            return 1;
        int c = 0;
        visited[source] = true;
        //recursion logic
        LinkedList<Edge> temp = g.adj.get(source);
        for(Edge e:temp)
        {
            if(visited[e.dest]==false)
```

```
                        c=c+countAllPaths(g,visited,e.dest,dest);
            }
            //backtracking
            visited[source] = false;
            return c;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(5);
            g.addUnDirectedEdge(0,1);
            g.addUnDirectedEdge(0,2);
            g.addUnDirectedEdge(1,3);
            g.addUnDirectedEdge(1,4);
            g.addUnDirectedEdge(2,3);
            g.addUnDirectedEdge(3,4);
            g.printGraph();
            System.out.println(Graph.countAllPaths(g,0,4));//4
            System.out.println(Graph.countAllPaths(g,0,2));//4
      }
}

print all paths from source to destination
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
```

```
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static void bfs(Graph g,int source){
            int v = g.v;
            boolean[] visited = new boolean[v];
            LinkedList<Integer> q = new LinkedList<Integer>();
            q.add(source);
            visited[source]=true;
            while(!q.isEmpty()){
                  int curr = q.remove();
                  System.out.print(curr+" ");
                  LinkedList<Edge> temp = g.adj.get(curr);
                  for(Edge e:temp){
                        if(visited[e.dest]==false){
                              visited[e.dest] = true;
                              q.add(e.dest);
                        }
                  }
            }
      }
      static void dfs(Graph g,int curr,boolean[] visited){
            System.out.print(curr+" ");
            visited[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp){
                  if(visited[e.dest]==false){
                        dfs(g,e.dest,visited);
                  }
            }
      }
      static void printAllPaths(Graph g,int source,int dest)
      {
            boolean[] visited = new boolean[g.v];
            Stack<Integer> path = new Stack<Integer>();
            printAllPaths(g,visited,source,dest,path);
      }
      static void printAllPaths(Graph g,boolean[] visited,int source,int
dest,Stack<Integer> path)
      {
            path.push(source);
```

```
        if(source==dest)
        {
            System.out.println(path);
            path.pop();
            return;
        }
        visited[source] = true;
        //recursion logic
        LinkedList<Edge> temp = g.adj.get(source);
        for(Edge e:temp)
        {
            if(visited[e.dest]==false)
                printAllPaths(g,visited,e.dest,dest,path);
        }
        //backtracking
        visited[source] = false;
        path.pop();
    }
}
class Test
{
    public static void main(String[] args)
    {
        Graph g = new Graph(5);
        g.addUnDirectedEdge(0,1);
        g.addUnDirectedEdge(0,2);
        g.addUnDirectedEdge(1,3);
        g.addUnDirectedEdge(1,4);
        g.addUnDirectedEdge(2,3);
        g.addUnDirectedEdge(3,4);
        g.printGraph();
        Graph.printAllPaths(g,0,4);
        Graph.printAllPaths(g,0,2);
    }
}
```

```
C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=> (0, 1) (3, 1) (4, 1)
vertex 2 is connected to=> (0, 1) (3, 1)
vertex 3 is connected to=> (1, 1) (2, 1) (4, 1)
vertex 4 is connected to=> (1, 1) (3, 1)
[0, 1, 3, 4]
[0, 1, 4]
[0, 2, 3, 1, 4]
[0, 2, 3, 4]
[0, 1, 3, 2]
[0, 1, 4, 3, 2]
[0, 2]
```

```
Detect cycle in an undirected graph
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ex1:
---
```

```java
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static boolean detectCycle(Graph g){
            boolean[] visited = new boolean[g.v];
            for(int i=0;i<g.v;i++){
                  if(!visited[i]){
                        if(detectCycleUtil(g,visited,i,-1))
                              return true;
                  }
            }
            return false;
      }
      static boolean detectCycleUtil(Graph g,boolean[] visited,int
curr,int parent)
      {
            visited[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e : temp){
                  if(!visited[e.dest]){//case1
                        if(detectCycleUtil(g,visited,e.dest,curr))
                              return true;
                  }
                  else if(visited[e.dest] && e.dest!=parent)//case2
                        return true;
                  //case3: do nothing
            }
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            g.addUnDirectedEdge(0,1);
            g.addUnDirectedEdge(0,2);
            g.addUnDirectedEdge(1,2);
            g.addUnDirectedEdge(2,3);
            g.printGraph();
            System.out.println(Graph.detectCycle(g));//true
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=> (0, 1) (2, 1)
vertex 2 is connected to=> (0, 1) (1, 1) (3, 1)
vertex 3 is connected to=> (2, 1)
true
```

Ex2:

```
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static boolean detectCycle(Graph g){
            boolean[] visited = new boolean[g.v];
            for(int i=0;i<g.v;i++){
                  if(!visited[i]){
                        if(detectCycleUtil(g,visited,i,-1))
                              return true;
                  }
            }
            return false;
      }
      static boolean detectCycleUtil(Graph g,boolean[] visited,int
curr,int parent)
      {
            visited[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e : temp){
                  if(!visited[e.dest]){//case1
                        if(detectCycleUtil(g,visited,e.dest,curr))
                              return true;
                  }
                  else if(visited[e.dest] && e.dest!=parent)//case2
                        return true;
                  //case3: do nothing
            }
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            g.addUnDirectedEdge(0,1);
            g.addUnDirectedEdge(0,2);
            g.addUnDirectedEdge(2,3);
            g.printGraph();
            System.out.println(Graph.detectCycle(g));//false
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=> (0, 1)
vertex 2 is connected to=> (0, 1) (3, 1)
vertex 3 is connected to=> (2, 1)
false
```

```
Detect cycle in an directed graph
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ex1:
----
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
```

```
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static boolean isCycle(Graph g){
            boolean[] visited = new boolean[g.v];
            boolean[] stack = new boolean[g.v];
            for(int i=0;i<g.v;i++){
                  if(!visited[i]){
                        if(isCycleUtil(g,visited,stack,i))
                              return true;
                  }
            }
            return false;
      }
      static boolean isCycleUtil(Graph g,boolean[] visited,boolean[]
stack,int curr)
      {
            visited[curr] = true;
            stack[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp)
            {
                  if(stack[e.dest])
                        return true;
                  if(!visited[e.dest] &&
isCycleUtil(g,visited,stack,e.dest))
                        return true;
            }
            stack[curr] = false;//backtracking
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            g.addDirectedEdge(0,1);
            g.addDirectedEdge(0,2);
            g.addDirectedEdge(2,3);
            g.addDirectedEdge(1,3);
            g.printGraph();
            System.out.println(Graph.isCycle(g));//false
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=> (3, 1)
vertex 2 is connected to=> (3, 1)
vertex 3 is connected to=>
false
```

Ex2:

```
import java.util.*;
class Graph{
      static class Edge{
            int dest;
            int cost;
            Edge(int dest,int cost){
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static boolean isCycle(Graph g){
            boolean[] visited = new boolean[g.v];
            boolean[] stack = new boolean[g.v];
            for(int i=0;i<g.v;i++){
                  if(!visited[i]){
                        if(isCycleUtil(g,visited,stack,i))
                              return true;
                  }
            }
            return false;
      }
      static boolean isCycleUtil(Graph g,boolean[] visited,boolean[]
stack,int curr)
      {
            visited[curr] = true;
            stack[curr] = true;
            LinkedList<Edge> temp = g.adj.get(curr);
            for(Edge e:temp)
            {
                  if(stack[e.dest])
                        return true;
                  if(!visited[e.dest] &&
isCycleUtil(g,visited,stack,e.dest))
                        return true;
            }
            stack[curr] = false;//backtracking
            return false;
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            g.addDirectedEdge(0,1);
            g.addDirectedEdge(0,2);
            g.addDirectedEdge(2,3);
            g.addDirectedEdge(3,0);
            g.printGraph();
            System.out.println(Graph.isCycle(g));//true
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 1) (2, 1)
vertex 1 is connected to=>
vertex 2 is connected to=> (3, 1)
vertex 3 is connected to=> (0, 1)
true
```

Shortest Path from source to all vertices

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import java.util.*;
class Graph{
      static class Edge{
            int source;
            int dest;
            int cost;
            Edge(int source,int dest,int cost){
                  this.source = source;
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(src,dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(src,dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
```

```
                  System.out.println();
            }
      }
      static class Pair implements Comparable<Pair>
      {
            int n;
            int path;
            public Pair(int n,int path){
                  this.n = n;
                  this.path = path;
            }
            public int compareTo(Pair p){
                  return this.path - p.path;
            }
      }
      static void dijkstra(Graph g,int source){
            int[] dist = new int[g.v];
            for(int i=0;i<g.v;i++){
                  if(i!=source)
                        dist[i] = Integer.MAX_VALUE;
            }
            //logic
            boolean[] visited = new boolean[g.v];
            PriorityQueue<Pair> pq = new PriorityQueue<>();
            pq.add(new Pair(source,0));
            while(!pq.isEmpty()){
                  Pair curr = pq.remove();
                  if(!visited[curr.n]){
                        visited[curr.n] = true;
                        LinkedList<Edge> temp = g.adj.get(curr.n);
                        for(Edge e:temp){
                              int u = e.source;
                              int v = e.dest;
                              int wt = e.cost;
                              if(dist[u]+wt <dist[v]){
                                    dist[v] = dist[u]+wt;
                                    pq.add(new Pair(v,dist[v]));
                              }
                        }
                  }
            }

            for(int i=0;i<g.v;i++)
                  System.out.print(dist[i]+" ");
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(6);
            g.addDirectedEdge(0,1,2);
            g.addDirectedEdge(0,2,4);
            g.addDirectedEdge(1,2,1);
            g.addDirectedEdge(1,3,7);
            g.addDirectedEdge(2,4,3);
            g.addDirectedEdge(3,5,1);
            g.addDirectedEdge(4,3,2);
            g.addDirectedEdge(4,5,5);
            g.printGraph();
            Graph.dijkstra(g,0);
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 2) (2, 4)
vertex 1 is connected to=> (2, 1) (3, 7)
vertex 2 is connected to=> (4, 3)
vertex 3 is connected to=> (5, 1)
vertex 4 is connected to=> (3, 2) (5, 5)
vertex 5 is connected to=>
0 2 3 8 6 9
```

```
Bellman Ford Algorithm:
~~~~~~~~~~~~~~~~~~~~~~~
```

Shortest path from the source vertex to all vertices (negative edges/weights).

```
import java.util.*;
class Graph{
      static class Edge{
            int source;
            int dest;
            int cost;
            Edge(int source,int dest,int cost){
                  this.source = source;
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
      void addDirectedEdge(int src,int dest,int cost){
            Edge edge = new Edge(src,dest,cost);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest,int cost){
            addDirectedEdge(src,dest,cost);
            addDirectedEdge(dest,src,cost);
```

```
      }
      void addDirectedEdge(int src,int dest){
            Edge edge = new Edge(src,dest,1);
            adj.get(src).add(edge);
      }
      void addUnDirectedEdge(int src,int dest){
            addDirectedEdge(src,dest,1);
            addDirectedEdge(dest,src,1);
      }
      void printGraph(){
            for(int i=0;i<v;i++){
                  LinkedList<Edge> temp = adj.get(i);
                  System.out.print("vertex "+i+" is connected to=> ");
                  for(Edge e : temp){
                        System.out.print("("+e.dest+", "+e.cost+") ");
                  }
                  System.out.println();
            }
      }
      static class Pair implements Comparable<Pair>
      {
            int n;
            int path;
            public Pair(int n,int path){
                  this.n = n;
                  this.path = path;
            }
            public int compareTo(Pair p){
                  return this.path - p.path;
            }
      }
      static void bellManFord(Graph g,int source){
            int dist[] = new int[g.v];
            for(int i=0;i<dist.length;i++){
                  if(i!=source)
                        dist[i] = Integer.MAX_VALUE;
            }
            for(int i=0;i<g.v-1;i++){
                  for(int j=0;j<g.v;j++){
                        LinkedList<Edge> temp = g.adj.get(j);
                        for(Edge e :temp){
                              int u = e.source;
                              int v = e.dest;
                              int wt = e.cost;
                              if(dist[u] != Integer.MAX_VALUE &&
dist[u]+wt < dist[v])
                                    dist[v] = dist[u] + wt;
                        }
                  }
            }
            for(int i=0;i<dist.length;i++)
                  System.out.print(dist[i]+" ");
      }
}
```

```
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(5);
            g.addDirectedEdge(0,1,2);
            g.addDirectedEdge(0,2,4);
            g.addDirectedEdge(1,2,-4);
            g.addDirectedEdge(2,3,2);
            g.addDirectedEdge(3,4,4);
            g.addDirectedEdge(4,1,-1);
            g.printGraph();
            Graph.bellManFord(g,0);
      }
}
```

```
C:\test>java Test
vertex 0 is connected to=> (1, 2) (2, 4)
vertex 1 is connected to=> (2, -4)
vertex 2 is connected to=> (3, 2)
vertex 3 is connected to=> (4, 4)
vertex 4 is connected to=> (1, -1)
0 2 -2 0 4
```

Minimum spanning tree (MST)
\~~~~~~~~~~~~~~~~~~~~~~~~~~

A minimum cost spanning tree or minimum weight spanning tree MST is asubset of the edges of a connected edge weighted undirected graph that connects all the vertices together without any cycle and min cost should be there.

Ex:

```
import java.util.*;
class Graph{
      static class Edge{
            int source;
            int dest;
            int cost;
            Edge(int source,int dest,int cost){
                  this.source = source;
                  this.dest = dest;
                  this.cost = cost;
            }
      }
      int v;
      static LinkedList<LinkedList<Edge>> adj;
      Graph(int v){
            this.v = v;
            adj = new LinkedList<LinkedList<Edge>>();
            for(int i=0;i<v;i++)
                  adj.add(new LinkedList<Edge>());
      }
```

```java
void addDirectedEdge(int src,int dest,int cost){
      Edge edge = new Edge(src,dest,cost);
      adj.get(src).add(edge);
}
void addUnDirectedEdge(int src,int dest,int cost){
      addDirectedEdge(src,dest,cost);
      addDirectedEdge(dest,src,cost);
}
void addDirectedEdge(int src,int dest){
      Edge edge = new Edge(src,dest,1);
      adj.get(src).add(edge);
}
void addUnDirectedEdge(int src,int dest){
      addDirectedEdge(src,dest,1);
      addDirectedEdge(dest,src,1);
}
void printGraph(){
      for(int i=0;i<v;i++){
            LinkedList<Edge> temp = adj.get(i);
            System.out.print("vertex "+i+" is connected to=> ");
            for(Edge e : temp){
                  System.out.print("("+e.dest+", "+e.cost+") ");
            }
            System.out.println();
      }
}
static class Pair implements Comparable<Pair>
{
      int n;
      int cost;
      public Pair(int n,int cost){
            this.n = n;
            this.cost = cost;
      }
      public int compareTo(Pair p){
            return this.cost - p.cost;
      }
}
static void primsMcst(Graph g)
{
      boolean[] visited = new boolean[g.v];
      PriorityQueue<Pair> pq = new PriorityQueue<>();
      pq.add(new Pair(0,0));
      int finalcost = 0;
      while(!pq.isEmpty()){
            Pair curr = pq.remove();
            if(!visited[curr.n]){
                  visited[curr.n] = true;
                  finalcost = finalcost+curr.cost;
                  LinkedList<Edge> temp = g.adj.get(curr.n);
                  for(Edge e:temp){
                        pq.add(new Pair(e.dest,e.cost));
                  }
            }
            }
            System.out.println(finalcost);
      }
}
class Test
{
      public static void main(String[] args)
      {
            Graph g = new Graph(4);
            g.addDirectedEdge(0,1,10);
            g.addDirectedEdge(0,2,15);
            g.addDirectedEdge(0,3,30);
            g.addDirectedEdge(1,3,40);
            g.addDirectedEdge(2,3,50);
            g.printGraph();
            Graph.primsMcst(g);
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
vertex 0 is connected to=> (1, 10) (2, 15) (3, 30)
vertex 1 is connected to=> (3, 40)
vertex 2 is connected to=> (3, 50)
vertex 3 is connected to=>
55
```

```
Krushkals algorithm:
--------------------
import java.util.*;
class Test
{
      static class Edge implements Comparable<Edge>{
            int source;
            int dest;
            int cost;
            Edge(int source,int dest,int cost){
                  this.source = source;
                  this.dest = dest;
                  this.cost = cost;
            }
            public int compareTo(Edge e2){
                  return this.cost - e2.cost;
            }
      }
      static void createGraph(ArrayList<Edge> edges)
      {
            edges.add(new Edge(0,1,10));
            edges.add(new Edge(0,2,15));
            edges.add(new Edge(0,3,30));
            edges.add(new Edge(1,3,40));
            edges.add(new Edge(2,3,50));
      }
      static int n = 4;
      static int par[] = new int[n];
      static int rank[] = new int[n];
      public static void init(){
            for(int i=0;i<n;i++)
                  par[i] = i;
      }
      public static int find(int x){
            if(par[x] == x)
                  return x;
            return par[x] = find(par[x]);
      }
      public static void union(int a,int b){
            int parA = find(a);
            int parB = find(b);
            if(rank[parA]==rank[parB]){
                  par[parB]=parA;
                  rank[parA]++;
            }
            else if(rank[parA]<rank[parB])
                  par[parA] = parB;
            else
                  par[parB] = parA;
      }
      public static void krushkals(ArrayList<Edge> edges,int v)
      {
            init();
            Collections.sort(edges);
            int mstCost = 0;
            int count = 0;
            for(int i=0;count<v-1;i++){
                  Edge e = edges.get(i);
                  int parA = find(e.source);
                  int parB = find(e.dest);
                  if(parA!=parB)
                  {
                        union(e.source,e.dest);
                        mstCost = mstCost + e.cost;
                        count++;
                  }
            }
            System.out.println(mstCost);
      }
      public static void main(String[] args)
      {
            int v = 4;
            ArrayList<Edge> edges = new ArrayList<>();
            createGraph(edges);
            krushkals(edges,v);
      }
}

C:\test>javac Test.java
```

```
C:\test>java Test
55
```

Priority Queues / Heaps

introduction:

1) it is also know as Heaps
2) items are inserted at end and removed from begining.
3) PQ is logical ordering of objects based on priority.
4) when we add an object in to PQ, reordering must be required (heapify).
5) when we remove an object from PQ, reordering must be required (heapify).

Ex:

Prims algorithm
Rank processing
student admission based on percentage
etc

6) PQ is implemented based on heaps
7) It is implemented by using arrays only.

predefined implementation of PQ:

Ex1:

Sort the given integer values based on asc order

```
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            PriorityQueue<Integer> pq = new PriorityQueue<>();
            pq.add(3);
            pq.add(4);
            pq.add(1);
            pq.add(7);
            while(!pq.isEmpty()){
                  System.out.println(pq.peek());//first most element
                  pq.remove();//first element will be removed
            }
      }
}
C:\test>javac Test.java

C:\test>java Test
1
3
4
7
```

Ex2:

sort integer values based on desc order using PQ

```
import java.util.*;

class Test
{
      public static void main(String[] args)
      {
            PriorityQueue<Integer> pq = new
PriorityQueue<>(Collections.reverseOrder());
            pq.add(3);
            pq.add(4);
            pq.add(1);
            pq.add(7);
            while(!pq.isEmpty()){
                  System.out.println(pq.peek());//first most element
                  pq.remove();//first element will be removed
            }
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
7
4
3
1
```

Ex3: sort the students based on htno in asc order

```
import java.util.*;

class Student implements Comparable<Student>{
      int htno;
      String name;
      Student(int htno,String name){
            this.htno = htno;
            this.name = name;
      }
      public String toString(){
            return "["+this.htno+", "+this.name+"]";
      }
      public int compareTo(Student temp){
            return this.htno - temp.htno;
      }
}
class Test
{
      public static void main(String[] args)
      {
```

```
		PriorityQueue<Student> pq = new PriorityQueue<>();
		pq.add(new Student(1,"Prakash"));
		pq.add(new Student(4,"Raju"));
		pq.add(new Student(2,"Kiran"));
		pq.add(new Student(3,"Karan"));
		pq.add(new Student(9,"Abhi"));
		while(!pq.isEmpty()){
			System.out.println(pq.peek());//first most element
			pq.remove();//first element will be removed
		}
	}
}
```

```
C:\test>javac Test.java

C:\test>java Test
[1, Prakash]
[2, Kiran]
[3, Karan]
[4, Raju]
[9, Abhi]
```

Ex4: sort the students based on htno in desc order

```
import java.util.*;

class Student implements Comparable<Student>{
	int htno;
	String name;
	Student(int htno,String name){
		this.htno = htno;
		this.name = name;
	}
	public String toString(){
		return "["+this.htno+", "+this.name+"]";
	}
	public int compareTo(Student temp){
		return this.htno - temp.htno;
	}
}
class Test
{
	public static void main(String[] args)
	{
		PriorityQueue<Student> pq = new 
PriorityQueue<>(Collections.reverseOrder());
		pq.add(new Student(1,"Prakash"));
		pq.add(new Student(4,"Raju"));
		pq.add(new Student(2,"Kiran"));
		pq.add(new Student(3,"Karan"));
		pq.add(new Student(9,"Abhi"));
		while(!pq.isEmpty()){
			System.out.println(pq.peek());//first most element
```

```
                  pq.remove();//first element will be removed
            }
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
[9, Abhi]
[4, Raju]
[3, Karan]
[2, Kiran]
[1, Prakash]
```

Heap

* It is a binary tree (at most two children)
* it is complete binary tree
  ==> all levels are completely filled except last level
  ==> filling should be done from left to right
* heap order property
  ==> children values >= parent values ---> minHeap ASC
  ==> children values <= parent values ---> maxHeap DESC

we can't represent heap tree in class list format ---> we will use array

Node at position x

left(x) ----> 2x+1
right(x) ---> 2x+2

min heap ---> Root Node must be smallest value
max heap ---> Root Node must be larger value

Insertion of an element in heap

step1: add the element at end
step3: fix heap (normal method)

Ex:

```
            public void add(int data){
                  //1. add the data at end
                  arr.add(data);
                  //2. fix heap
                  int ci = arr.size()-1;
                  int pi = (ci-1)/2;
                  while(arr.get(ci)<arr.get(pi)){
                        int temp = arr.get(ci);
                        arr.set(ci,arr.get(pi));
                        arr.set(pi,temp);
                        ci = pi;
                        pi = (ci-1)/2;
            }
        }
```

deletetion in heap

```
import java.util.*;

class Test
{
    static class Heap{
        ArrayList<Integer> arr = new ArrayList<>();
        public void add(int data){
            //1. add the data at end
            arr.add(data);
            //2. fix heap
            int ci = arr.size()-1;
            int pi = (ci-1)/2;
            while(arr.get(ci)<arr.get(pi)){
                int temp = arr.get(ci);
                arr.set(ci,arr.get(pi));
                arr.set(pi,temp);
                ci = pi;
                pi = (ci-1)/2;
            }
        }
        public int remove(){
            int data = arr.get(0);
            //step1: swap first and last
            int temp = arr.get(0);
            arr.set(0,arr.get(arr.size()-1));
            arr.set(arr.size()-1,temp);
            //step2: delete last
            arr.remove(arr.size()-1);
            //step3: heapify
            heapify(0);
            return data;
        }
        public void heapify(int index)
        {
            int left = 2*index + 1;
            int right = 2*index + 2;
            int mi = index;
            if(left<arr.size() && arr.get(mi)>arr.get(left))
                mi = left;
            if(right<arr.size() && arr.get(mi)>arr.get(right))
                mi = right;
            if(index!=mi){
                int temp = arr.get(index);
                arr.set(index,arr.get(mi));
                arr.set(mi,temp);
                heapify(mi);
            }
        }
```

```
            public int peek(){
                  return arr.get(0);
            }
            public boolean isEmpty(){
                  return arr.size()==0;
            }
      }
      public static void main(String[] args)
      {
            Heap h = new Heap();
            h.add(3);
            h.add(4);
            h.add(1);
            h.add(5);
            while(!h.isEmpty()){
                  System.out.println(h.peek());
                  h.remove();
            }
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
1
3
4
5
```

```
heap sort:
----------
==> ascending order -----> max heap
==> descending order ----> min heap
```

```
import java.util.*;

class Test
{
      public static void heapSortAsc(int[] a){
            //step1: to build max heap
            int n=a.length;
            for(int i=n/2;i>=0;i--)
                  heapify(a,i,n);
            //step2: push largest element at end
            for(int i=n-1;i>0;i--){
                  //swap
                  int temp;
                  temp=a[0];
                  a[0] = a[i];
                  a[i] = temp;
                  heapify(a,0,i);
            }
      }
      public static void heapify(int[] a,int i,int n){
            int left = 2*i+1;
            int right = 2*i+2;
            int maxIndex = i;
            if(left<n && a[left] > a[maxIndex])
                  maxIndex = left;
            if(right<n && a[right] > a[maxIndex])
                  maxIndex = right;
            if(maxIndex!=i)
            {
                  //swap
                  int temp = a[i];
                  a[i] = a[maxIndex];
                  a[maxIndex] = temp;
                  heapify(a,0,i);
            }
      }
      public static void main(String[] args)
      {
            int[] a = {1,2,4,5,3};
            System.out.println(Arrays.toString(a));//[1,2,4,5,3]
            heapSortAsc(a);
            System.out.println(Arrays.toString(a));//[1,2,3,4,5]
      }
}

C:\test>javac Test.java

C:\test>java Test
[1, 2, 4, 5, 3]
[1, 2, 3, 4, 5]

Ex:
---
import java.util.*;

class Test
{
      public static void heapSortDesc(int[] a){
            //step1: to build min heap
            int n=a.length;
            for(int i=n/2;i>=0;i--)
                  heapify(a,i,n);
            //step2: push smallest element at end
            for(int i=n-1;i>0;i--){
                  //swap
                  int temp;
                  temp=a[0];
                  a[0] = a[i];
                  a[i] = temp;
                  heapify(a,0,i);
            }
      }
      public static void heapify(int[] a,int i,int n){
            int left = 2*i+1;
            int right = 2*i+2;
            int minIndex = i;
            if(left<n && a[left] < a[minIndex])
                  minIndex = left;
            if(right<n && a[right] < a[minIndex])
                  minIndex = right;
            if(minIndex!=i)
            {
                  //swap
                  int temp = a[i];
                  a[i] = a[minIndex];
                  a[minIndex] = temp;
                  heapify(a,0,i);
            }
      }
      public static void main(String[] args)
      {
            int[] a = {1,2,4,5,3};
            System.out.println(Arrays.toString(a));//[1,2,4,5,3]
            heapSortDesc(a);
            System.out.println(Arrays.toString(a));//[1,2,3,4,5]
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
[1, 2, 4, 5, 3]
[5, 4, 3, 2, 1]
```

Divide and Conquer Algs:

==> In D and C, we will break the problem into sub-problems.
==> Then solve sub-problems independently.
==> Combine the solutions to get solution for main problems.
==> we have to solve this sub-problems by using recursion.
==> This D and C has three forms

1) Divide -----> smaller instances
2) Conquer ----> solve sub problems recursively
3) Combine ----> combine the solutions

Ex:

Merge Sort
Binary Search
Quick sort
Towers of Hanoi etc

Dynamic Programming:

=> almost it is same like recursion.

=> recursion with optimized solution is nothing but DP.

Ex: Fibonacci Sequence General Method

```
import java.util.*;

class Test
{
	static int c1 = 0;
	public static int fibGeneral(int n){
		c1++;
		if(n==0||n==1)
			return n;
		else
			return fibGeneral(n-1)+fibGeneral(n-2);
	}
	public static void main(String[] args)
	{
		//0, 1, 1, 2, 3, 5, 8, .....
		int n = 6;
		System.out.println(fibGeneral(n));//8
		System.out.println("Number of calls: "+c1);
	}
}
```

```
C:\test>javac Test.java

C:\test>java Test
8
Number of calls: 25
```

Ex: Fibonacci Sequence Modified Method

```
import java.util.*;

class Test
{
	static int c1 = 0;
	static int c2 = 0;
	public static int fibGeneral(int n){
		c1++;
		if(n==0||n==1)
			return n;
		else
			return fibGeneral(n-1)+fibGeneral(n-2);
	}
	public static int fibModified(int n,int[] f){
		c2++;
		if(n==0||n==1)
			return n;
		if(f[n]!=0)
			return f[n];
		f[n] = fibModified(n-1,f)+fibModified(n-2,f);
```

```
            return f[n];
      }
      public static void main(String[] args)
      {
            //0, 1, 1, 2, 3, 5, 8, .....
            int n = 6;
            System.out.println(fibGeneral(n));//8
            System.out.println("Number of calls: "+c1);//25
            int[] f = new int[n+1];
            System.out.println(fibModified(n,f));//8
            System.out.println("Number of calls: "+c2);//
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
8
Number of calls: 25
8
Number of calls: 11
```

What is dynamic programming?

Dynamic programming is optimized recursion process.

How to identify dynamic programming is applicable for a problem?

1) optimal problem
2) some choice is given (multiple branches)

there are two ways are there to solve dynamic programming

1) memoization (top down) (recursion, subproblems (reuse))
2) tabulation (bottom up) (intialization, filling an array)

application11: fibanocci of the given number

```
import java.util.*;

class Test
{
      public static int fibGeneral(int n){
            if(n==0||n==1)
                  return n;
            else
                  return fibGeneral(n-1)+fibGeneral(n-2);
      }
      public static int fibMemoization(int n,int[] f){
            if(n==0||n==1)
                  return n;
            if(f[n]!=0)
                  return f[n];
            f[n] = fibMemoization(n-1,f)+fibMemoization(n-2,f);
            return f[n];
      }
      public static int fibTabulation(int n){
            int dp[] = new int[n+1];
            dp[0] = 0;
            dp[1] = 1;
            for(int i=2;i<=n;i++){
                  dp[i] = dp[i-1]+dp[i-2];
            }
            return dp[n];
      }
      public static void main(String[] args)
      {
            //0, 1, 1, 2, 3, 5, 8, .....
            int n = 6;
            System.out.println(fibGeneral(n));//8
            int[] f = new int[n+1];
            System.out.println(fibMemoization(n,f));//8
            System.out.println(fibTabulation(n));//8
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
8
8
8
```

```
application2: climbing stairs
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

count ways to reach the nth stair, the person can climb either 1 stair or 2 stair at a time.

```
import java.util.*;

class Test
{
      public static int countWaysGeneral(int n){
            if(n==0)
                  return 1;
            if(n<0)
                  return 0;
            return countWaysGeneral(n-1)+countWaysGeneral(n-2);
      }
      public static int countWaysMemoization(int n,int[] ways)
      {
            if(n==0)
                  return 1;
            if(n<0)
                  return 0;
            if(ways[n]!=-1)//already calculated
                  return ways[n];
```

```
            ways[n] = countWaysMemoization(n-
1,ways)+countWaysMemoization(n-2,ways);
            return ways[n];
      }
      public static int countWaysTabulation(int n){
            int[] dp = new int[n+1];
            dp[0] = 1;
            for(int i=1;i<=n;i++){
                  if(i==1)
                        dp[i] = dp[i-1];
                  else
                        dp[i] = dp[i-1]+dp[i-2];
            }
            return dp[n];
      }
      public static void main(String[] args)
      {
            //1, 1, 2, 3, 5,.....
            int n = 6;
            System.out.println(countWaysGeneral(n));//13
            int[] ways = new int[n+1];
            Arrays.fill(ways,-1);
            System.out.println(countWaysMemoization(n,ways));//13
            System.out.println(countWaysTabulation(n));//13
      }
}
```

```
C:\test>javac Test.java

C:\test>java Test
13
13
13
```

Time & Space complexities

complexity of an algorithm is the amount of time and space required to complete its execution.

Time Complexity T(n) -----> amount of time taken
Space Complexity S(n) ----> amount of space taken

Note: Space Complexity we are not required to consider. (High Level Programming->GC)

Asymptotic Notations or Asymptotic Analysis:

calculating running time of any algorithm in mathmatical units of computation is know as aymptotic notations.

Big-Oh Notation ----> O
Omega Notation -----> w
Theta Notation -----> 0

complexity analysis of an algorithm:

worst case complexity ------> Bigoh (O) ---> most commonly used notation
best case complexity -------> Omega (w)
average case complexity ----> Theta (0)

Growth of functions:

~~~~~~~~~~~~~~~~~~~~

O(1) constant time:

Algorithm will return on constant time.

Ex:

- Access nth element from an array
- Push and Pop operations on stack
- add and remove operations from queue
- accessing element from hashtable etc

O(n) linear time:

Execution time is directly proportional to input size.

Ex:

- search operation
- min element in an array
- max element in an array
- traversal of a tree
- etc

Logarithmic Time O(logn):

Algorithm is said to run in logarithm time. if the execution time of an alg is proportional to logarithm of input size.

Ex:

- binary search

Logarithmic Time O(nlogn):

Algorithm will run in n*logn time, if the execution of an algorithm is proportional to the product of input size and logarithm value of input size.

Ex:

- Merge Sort
- Quick Sort
- Heap Sort
- All Divide and Conquer etc

Quadratic Time O(n2):

An algorithm is said to run in quadratic time of an algorithm is proportional to square of input size.
~~~~~~~~~~~~~~~~~~~~

```
Ex:
      Bubble Sort
      Selection Sort
      Insertion Sort
      etc
```

```
Ex1:
----
import java.util.*;

class Test
{
      public static int m(int n){
            int c=0;
            for(int i=0;i<n;i++)
                  c++;
            return c;
      }
      public static void main(String[] args)
      {
            System.out.println("N=100, number of instructions is O(n):
"+m(100));
      }
}

C:\9pm>javac Test.java

C:\9pm>java Test
N=100, number of instructions is O(n): 100
```

```
Ex2:
----
import java.util.*;

class Test
{
      public static int m(int n){
            int c=0;
            for(int i=0;i<n;i++)
            {
                  for(int j=0;j<n;j++){
                        c++;
                  }
            }
            return c;
      }
      public static void main(String[] args)
      {
            System.out.println("N=100, number of instructions is O(n2):
"+m(100));//10000
      }
}
```

```
C:\9pm>java Test
N=100, number of instructions is O(n2): 10000
```

```
Ex3:
----
import java.util.*;

class Test
{
      public static int m(int n){
            int c=0;
            for(int i=0;i<n;i++)
            {
                  for(int j=0;j<n;j++){
                        for(int k=0;k<n;k++){
                              c++;
                        }
                  }
            }
            return c;
      }
      public static void main(String[] args)
      {
            System.out.println("N=100, number of instructions is O(n3):
"+m(100));//1000000
      }
}
```

```
Ex4:
----
import java.util.*;

class Test
{
      public static int m(int n){
            int c=0;
            for(int i=n;i>0;i=i/2)
            {
                  c++;
            }
            return c;
      }
      public static void main(String[] args)
      {
            System.out.println("N=100, number of instructions is O(logn):
"+m(100));//7
      }
}
```

```
Ex5:
---
import java.util.*;

class Test
```

```
{
      public static int m(int n){
            int c=0;
            for(int i=1;i<=n;i=i*2)
            {
                  c++;
            }
            return c;
      }
      public static void main(String[] args)
      {
            System.out.println("N=100, number of instructions is O(logn): 
"+m(100));//7
      }
}
```

Greedy Method:

The main purpose of this method is to find optimal solution for the given problem

Ex1: coins

```
coins ---> [10, 5, 2, 1]
amount --> 52

solution1: 52x1=52           ----> 52 coins
solution2: 25x2+2x1=50+2 =52 ----> 27 coins
.
.
.
solutionx: 5x10+1x2 = 50+2 = 52 ----> 6 coins
```

```
import java.util.*;
class Test
{
      public static void sortDesc(int[] a){
            for(int i=0;i<a.length;i++){
                  for(int j=i+1;j<a.length;j++){
                        if(a[i]<a[j]){
                              int t=a[i];
                              a[i] = a[j];
                              a[j] = t;
                        }
                  }
            }
      }
      public static int minCoins(int[] coins,int amount)
      {
            int res = 0,j;
            sortDesc(coins);
            for(int i:coins){
                  if(i<=amount){
                        j = amount/i;
```

```
                res = res + j;
                amount = amount - (j*i);
            }
            if(amount==0)
                break;
        }
        return res;
    }
    public static void main(String[] args)
    {
        int[] coins = {5, 10, 2, 1};
        int amount = 52;
        System.out.println(minCoins(coins,amount));
    }
}
```

General Method for solving greedy method

```
void getOptimalSol()
{
    res = 0;
    while(all items are not considered)
    {
        i = selectAnItem;
        if(feasible(i))
         res = res + i;
    }
    return res;
}
```

Knapsack(bag) problem

items = [60, 100, 120]
weight = [10, 20, 30]
W = 50

AVL Trees
RED & BLACK Trees

syllabus:

01. introduction to dsa
02. sample algorithms and implementations
03. arrays data structure
04. CRUD operations on arrays (insert, access, update & delete)
05. One-D, Two-D(matrix) and Three-D array programs
06. string data structure
07. recursion and applications
08. sorting and searching algorithms
09. list data structures (SLL, DLL, CSLL & CDLL)
10. stack data structure
11. queue data structure
12. hashtable data strcture

13. tree data structure
14. priority queues and heaps
15. graphs
16. dynamic programming
17. divide and conquer
18. greedy algorithms
19. backtracking applications
20. complexities