# Algorithms - 01

## ✳ Traversing a Linear Array ✳

Step 1: Set K := LB.

Step 2: Repeat step 3 and 4 While K ≤ UB.

Step 3: Write LA[K].

Step 4: Set K := K+1

Step 5: Exit:-

### Implementation in Java :-

```
Public class Traversing {
Public static void main (String args[]){
 int LA[] = {2,4,6,8,0,9};
int LB=0, UB= LA.Length-1;
 int K=LB;
    While (K ≤ UB) {
System.out.print (LA[k]+" ");
    k++;
  }
 }
}
```

# Algorithms – 02

Inserting an element into a Linear Array:

Insert (LA[], N, J, P, Item).

Step 1: Set J := N.

Step 2: Repeat steps 3 and 4 while J ≥ P.

Step 3: Set LA[J+1] := LA[J].

Step 4: Set J := J - 1.

Step 5: Set LA[P] := Item.

Step 6: Exit.

## * Implementation in Java *

```
Public class Insert{
public static void main(String args[]){
 int LA[]={2,4,6,8,10,12,0,0};
 int item = 22;

int j=LA.Length-1, P=3;

   While (j>=P) {

  LA[j+1]=LA[j];
     j--;
}
   LA[P]=item;
For (int i=0; i< LA.Length-1; i++){

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

   }

   }

}.
```

# Algorithms:- 03

## Deleting an element from a Linear Array"

Delete (P, N, LA[], Item, J)

Step 1:- Set Item := LA[P]
Step 2:- Repeat
    Step 3 For J := P to N-1.

Step 3:- Set LA[J] := LA[J+1].
    End for Loop.

Step 4:- N := N-1.
Step 5:- Exit.

## Implementation:

```
public class detete{
public static void main (String args[]){
  int p = 3 , item = LA[P];
  For (int j=p; j< LA.Length-1; j++){
    LA[j] = LA[j+1]; }
For (int i=0; i<. LA.Length; i++){

System.out.print(LA[i]+"   ");
    }
    }
  }.
```

# Algorithm :- 04

## Insertion in Sorted - Array -

Insert (LA[], N, I, Item)

Step 1:- Set I := N.
Step 2:- Repeat step 3 and 4 while I ≥ 0 && LA[I] > Item.
Step 3:- Set LA[I+1] := LA[I].
Step 4:- Set I := I - 1.
End of step 2 Loop.
Step 5:- Set LA[I+1] := Item
Step 6:- Set N := N+1.
Step 7:- Exit.

## Implementation in Java

```
public class Insertion{
public static void main (String args[]){
int LA[]={10,20,30,40,50,60,70};
  int j=LA.Length-1;
  int item = 26;
  While (j>=0 && LA[j]>item){
    LA[j+1]=LA[j];
      j--;
  }
     LA[j+1]= item;
For (int i=0; i< LA.Length; i++){
System.out.print (LA[i]+ "  ");
    }
  }
}.
```

# ✳ Algorithm :- 05 ✳

## ✳ Linear Search ✳

Linear (DATA, N, Item, Loc)

Step 1 :- Set Data[N+1] := Item.

Step 2 :- Set Loc := 1.

Step 3 :- Repeat While Data[Loc] ≠ Item

Set Loc := Loc + 1.
[End of Loop]

Step 4: If Loc := N+1.

Write [Successful],
then
Set Loc := 0.
Write [unsuccessful].

Step 5 :-

Exit.

## Implementation in Java

```
public class Search {
public static void main (String args[])
 int LA[] = { 10, 20, 30, 40, 50, 60, 70, 80};

 int N = LA.Length-1, item = 100;

 LA[N+1] = item, Loc = 0;

 While (LA[Loc] := item) {
     Loc = Loc + 1;
  }

  if (Loc = N+1) {

System.out.println (" Item Found at Location;"+
      Loc);.

 }
   else
{
System.out.println (" Item Not Found");

   }

  }.
```

# Algorithm No: 06

## ✳ Binary Search ✳

Binary (LA[], LB, UB, Item, Loc).

Step 1:- Set Begging := LB, End := UB.
and mid = (Begging + End)/2.

Step 2:- Repeat Step 3 and Step 4
While Beg ≤ End.
and LA[mid] ≠ Item.

Step 3:- If Item < LA[mid], then

Set End := Mid - 1.
Else:
Set Begging := Mid + 1.
[End of if Structure].

Step 4:- Set Mid := (Beg + End)/2.
(End of while Loop).

Step 5:- If LA[Mid] = Item, then
Set Loc := Mid.
Else.
Set Loc := Null.

Step 6: Exit.

## Implementation in Java

```
Public class Binary Search {
Public static void main (String args[]) {
int LA[] = { 10, 20, 30, 40, 50, 60, 70, 80, 90};
int item = 50;
int Beg = LB , End = UB;
int Mid = ( Beg + End) /2 ;
While (Beg <= End && LA [Mid] != Item) {
    if (LA [Mid] < Item) {
Beg = Mid + 1;
    }
    else {
    End = Mid – 1 ;
    }
Mid = (Beg + End) /2 ;
}
 if (LA[Mid] == Item) {
```

```
System.out.print("Item found at Location:
    + Mid);

  }
     else
  {

System.out.println("Item not found");

  }

  }

}.
```

# Algorithm – 07

## Floor and Ceiling function:

### Floor function:.

For floor function take the input number and round down to the nearest integar.

### Ceiling function:-

For ceiling function the l'input number and round up to the nearest integar.

## Implementation in Java

```
Public class floorceiling-function {

Public Static int
        floor (double number) {
            return (int)
Math.floor (number): }

Public Static int
        Ceiling ( double number) {
            return (int)
Math.ceil (number); }.
Public Static void main (String args[]) {
        double number = 3.7;
System.out.println(" Floor " + number + " is " + floor(number));
System.out.println(" Ceiling " + number + " is " + (ceiling(number));
}
```