x = 597

size = 4 bytes

Array :-

Page No.

Date : 29/12/2023

Array is a group of or collection of similar types of data

* Array declaration :-

Syntax :- data type [] variable-name;

or

data type [] var-name;

or

datatype Var-name[];

EX :-

```
int [] x;
double [] y;
etc.
```

* Creation of array :-

→ Array can be created in two diff^m ways

(i) without new keyword

(ii) with new keyword

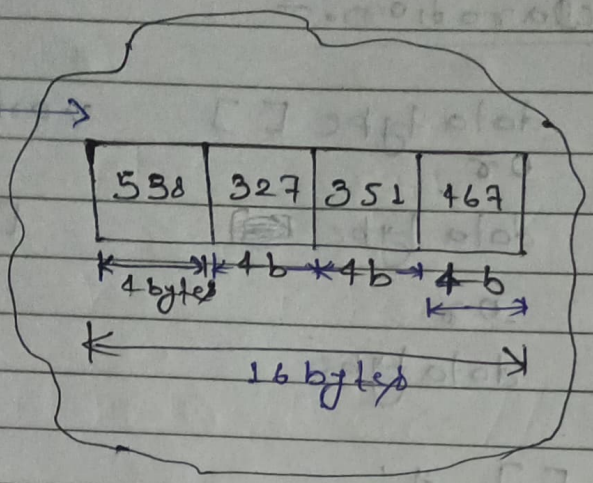
(i) without new keyword :- we can create an array directly by using curly braces as per the values provided inside curly braces in array object will be created inside heap area.

→ EX :-

```
int [] p = {588, 327, 351, 467};
```

Here an array object will be created internally at heap area. as there are four values inside curly braces so

four continues memory block will be created at heap area where values will be created.



Heap area

double[] k = {30.8, 48.7, 36.8, 46.5, 12.8};

String[] p = {"mohan", "soham", "john", "Delhi" "Rome"};

Char[] z = {'@', '+', '-', '*'};

DATE 02/01/2024

Array creation with new keyword:-

→ Array is a non primitive type data so it can be created by using new keyword.

→ When array is created by using new keyword then the default value will be stored at all the positions of array.

→ Later this values can be updated or changed.

Page No.
Date: 02/1/24

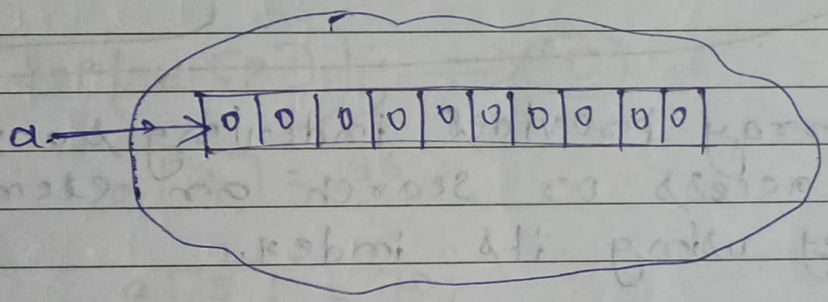
Note:- Array will be created in always heap area cause it is a non-primitive data.

Syntax:-

datatype [] var-name = new datatype[size];

Ex:-

int [] a = new int [10];



Heap area

* Important point about array:-

1) Array is fixed in size.
It's size can not be changed at run time.

2) Array provides support of a variable called length by which we can get the length of the array.

Ex:-

```
int [] a = {12, 15, 16, 18, 20};
Sop(a.length);   // Array has variable
```

```
int[] b = new int[12];
    SOP(b.length);

String[] c = {"abc","pqr","defg","kjh"};
    SOP(c.length);
```

```
String k = "mohan is here";
    SOP(k.length);   // String has as method
```

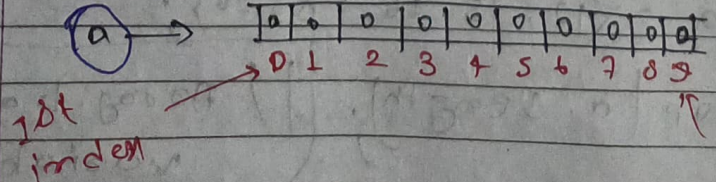
3) Array provides indexing so we can update, access or search an element of array by using its index.

4) The indexing of array starts from 0 and subsequently it increases. the index value also represents how far it is from the first element.

→ The last index will be always one less then from total length

EX:-

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

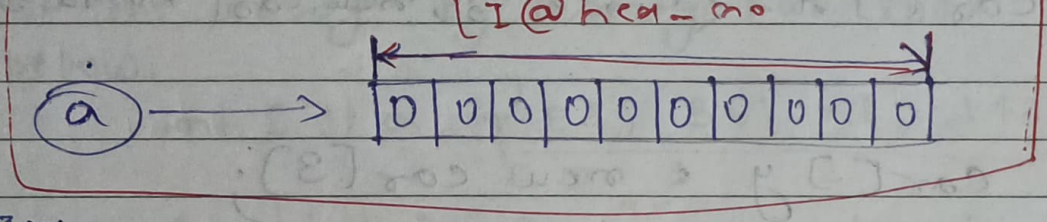



```
last.index = a.length-1;
```

Page No.
Date : 2/1/2024

(5) On array reference we will get memory block reference where array object is created. To get the value of array we have to use indexing

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



Ex:-

```
sop( a[2] ); => 0

a[4] = 36;

a[5] = 70;

sop ( a[4]); => 36

sop ( a[5]); => 70
```

sop(a); => Array Obj. address [I @ ---

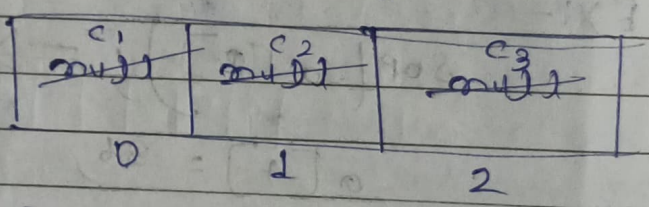
Page No.
Date : 2/1/24

(6) By using array we can store primitive or non primitive types of data.

```
Car c1 = new Car("maruti", 356892, "Blue");
Car c2 = new Car("tata", 5689512, "white");
Car c3 = new Car("Audi", 1234567, "black");

Car[] x = {c1, c2, c3, new Car("Honda", 3456782, "red")};

Car[] y = new Car[3];
```




```
y[0] = c1;
y[1] = c2;
y[2] = c3;
```

3/1/2024

(7) Whenever we will try to add or remove any element which is not in the range of array then it will give an exception at run time called array index out of bound exception.

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

a[10] = 950;

Sop(a[10]);
```

// Array index out of bound exception

Page No.
Date : 3/1/24

(8) We can access the each elements of an array by using loops. two commonly use loops are:-

(i) For loop
(ii) For each loop / Enhanced for loop

(i) For loop :- we can access each elements of the array by using for loop as below.

```
int [] a = {58, 36, 62, 148};
           0   1   2   3
for(int i = 0; i < 4; i < a.length; i++)
{
    sop(a[i]);
}
```

a[3]

O/P => 58
36
62
148

(ii) For each loop :- for each loop is used only where we have to iterate a group of elements such as Array, ArrayList, LinkedLists etc.

Syntax of for each loop :-

```
for(datatype variable_name : Array/collection ref.)
{
    // For each loop

}
```

Page No.
Date : 3/1/23

Ex:- int[] a = {58, 36, 62, 148};

0	1	2	3
58	36	62	148

Curcer →

```
for(int x : a)
{
    SOP(x);
}
```

```
Ob => 58
      36
      62
      148
```

(9) To build dynamic program we have to take length input and all the elements values from user at our time.

Ex:-
```
SOP("Enter the size");
int n = Sc.nextInt();

int[] a = new int[n];
for(int i=0; i<a.length; i++)
{
SOP("Enter the element number:"+(i
    a[i] = Sc.nextInt();
}
SOP("All the Elements are:");
for(int x: a)
{
    SOP(x);
}
```

Page No.
Date: 3/1/24

10 Array is fixed in size so array is recommended to use when we known in advance how many elements are to be stored.

→ When we don't known the no. of elements to be stored then in that case advance data structure should be used which comes under collection framework.

Q Take one array and access all the elements which is available at odd index of the array.

→ int [] a = {12, 18, 30, 60, 40, 90, 100};
0 1 2 3 4 5 6

By
→ For loop:-

```
for(int i=0 ; i<a.length ; i++)
{
    if(i%2!=0) or (i%2==1)
    {
        SOP(a[i]);
    }
}
```

By
→ For each loop:-

```
int count = 0;
for(int x : a)
{
    if(count%2==1)
    {
        SOP(x);
    }
    count++;
}
```

Page No.
Date : 3/1/24

Q (2) Take one array and access all the even elements of array.

```
int[] a = {12, 18, 31, 60, 47, 97, 120};
```

→ By for loop!-

```
for (int i=0; i<a.length; i++)
{
    if (a[i] % 2 == 0)
    {
        sop(a[i]);
    }
}
```

→ By for each loop

```
for(int x : a)
{
    if (x % 2 == 0)
    {
        sop(x);
    }
}
```

Page No.
Date : 9/1/24

HW

(3) WAP to access and count all the tree digits number in the array.

(4) WAP to print and count all the elements of array which is divided by Seven.

(3) Soln:- int [ ]a = {234, 456, 678, 789};

~~Public Static~~
class ThreeDigitNumbers
{
Public Static void main (String [ ] args)
{
int [ ] a = {234, 456, 678, 789};
int count = 0;
for (int x : a)
{
if (x >= 100 && x <= 999)
{
count++;
}
}

System.out.println ("The count of three-digit number is :" + count);
}
}

Page No.
Date : 4/1/24

Q1. WAP to print sum and average of all the elements of array.

```
int [] a = {12, 15, 20, 13, 40};
double int Sum = 0;
for (int p : a)
{
    Sum = Sum+p
}
SOP("Total Sum is :" +Sum);
SOP("Avg is :" + Sum/a.length);
```

Note:-

```
int[] a = {12, 15, 18, 20};
String b = "mohan is here";
SOP (a.length);
SOP (b.length());
```

Page No.
Date : 4/1/24

(2) WAJP to Print the Biggest element, Smallest element and their difference in the array.

```
int [] a = {12, 15, 20, 8, 40, 32};

int biggest = a[0];

~~int bigge~~
int Smallest = a[0];
for( int i = 1; i < a.length; i++ )
{
    if( a[i] > biggest )
        biggest = a[i];
    else if( a[i] < Smallest )
        Smallest = a[i];
}
SOP("Biggest is:" + Biggest);
SOP("Smallest is:" + Smallest);
```

Page No.
Date : 4/1/24

(5) ~~WAJP to print~~

(5) For the given array of strings. print and count all the strings which has even number of character.

```
String [] P = {'mohan', 'John', 'sohan', 'pqr'
                                        'abcd'};
int count = 0
for(int i = 0; i < P.length; i++)
{
    if(P[i].length % 2 == 0).
    {
        sop(P[i]);
        count++;
    }
}
sop("Total no of string are!" + count);
```

By for loop →

→ Now By for each loop:

```
int count = 0;
for(String k : P)
{
    if(k.length() % 2 == 0)
    {
        SOP(k);
        count++;
    }
}
sop(" Total String !" + count );
```

Page No.
Date : 4/1/24

(6) WAJP to print and count all the prime numbers of the array.

```
Public static void main(String[] args)
{
    int[] a = {23, 40, 17, 18, 28, 43};
    int count = 0;
    for(int x : a)
    {
        if(checkPrime(x))
        {
            Sop(x)
            count++;
        }
    }
    System.out.println("Total count is:" + count);
}
Public static boolean checkPrime(int n)
{
    for(i=2; i<=n/2; i++)
    {
        if(n%i==0)
            return false;
    }
    return true;
}
```

Page No.
Date : 4/1/24

(7) WAJP to reverse and print each element of the array.

Page No.
Date : 4/1/24

Q8. WAJP to swap two index valued of the array. → (with temp)

int [] a = {12, 15, 30, 60, 18, 19};

(1) (4)

```
int temp = a[1];
a[1] = a[4];
a[4] = temp;
```

→ Without temp | int [] a = {12, 15, 30, 60, 18, 19};

(1) (4)

a[1] 33 18

```
a[1] = a[1] + a[4];
a[4]   a[1] - a[4];
       33 - 18
       (15)
```

```
a[1] = a[1] - a[4];
       33 - 15
       (18)
```

Page No.
Date : 4/1/24

(9) WAJP to reverse and print the array.

```
int[] a = {12, 15, 20, 8, 40, 32};

for(int i = a.length - 1; i >= 0; i--)
{
    SOP(a[i]);
}
```

Page No.
Date : 4/1/24

Q (10) WAJP to check no is even or odd without using if else / conditional operator statement.

5387

```
int n = sc.nextInt();
String[] k = {"Even", "Odd"};
SOP(n + " is " + k[n/2]);
```

o/p :- 5387 is : Odd

Page No.
Date : 5/1/24

(11) WAJP to print the frequency of each element of the array.

int [ ] a = {12, 20, 18, 12, 16, 20, 12, 20, 12};

O/P =>
12 i/p => 4 times
20 i/p => 3 times
18 i/p => 1 times
16 i/p => 1 times

class Frequency
{
Public static void .

Public static void getFrequency(int[] a)
{ int n = a.length - 1
for(int i = 0; i < a.length; i++)
{ int count = 1;
for(int j = i+1; j <= n; j++)
{
if(a[i] == a[j])
{ count++;
a[j] = a[n];
n--;
j--;
}}
sop(a[i] + " i/p: => " + count + " times");
}

Page No.
Date : 8/1/24

(12) WAJP to Print each element of the array which has appeared only once in the array.

↳ [12, 18, 12, 30, 16, 40, 30]

o/p => 18
16
40

```
Public Static void PrintElement(int[] a)
{
    int n = a.length-1;
    for(int i=0, i<=n, i++)
    {
        int count = 1;
        for(int j = i+1; j<=n; j++)
        {
            count ++;
            a[j] = a[n];
            j--;   n--;
        }
        if(count == 1)
            SOP(a[i]);
    }
```

if (count > 1)

Page No.
Date :

Q 13 WAJP to Print each element of the array which has appeared more than once / which has dublicate values in the array.

if (count % 2 = 1)

Q14] WAP to Print all the Element which has appeared for max time in the array.

Page No.

Date : frequency and it's

15) WAJP to print the element which has appeared for the max^m time in the array

↳

[12, 18, 12, 30, 16, 40, 30, 12, 40, 12]

↳ o/p => (12) has appeared 4 times

```
Public Static void printElement(int[] a)
{
    int n = a.length - 1;
    int maxcount = 0;
    int target = 0;
    for(int i=0, i<=n, i++)
    {
        int count = 1;
        for(int j = i+1; j<=n; j++)
        {
            count++;
            a[i] = a[n];
            j--; n--;
        }
    if(maxcount < count)
    {
        maxcount = count;
        target = a[i];
    }
    SOP(target "is :" + count + "times");
}
```

Page No.
Date :

Q WAP to Print the index as well as the value of first non-repeated element

[12, 30, 12, 30, 16, 40, 30, 15, 12, 30, 12]

O/P ⇒ 16 and its index is : 4

Page No.
Date : 02/1/21

(16) WAP to Print Biggest and Second biggest element of the array.

{ ~~1, 2, 7, 18, 30, 21, 1~~ }

int [ ] a → [18, 20, 16, 24, 30]
indexes: 0, 1, 2, 3, 4

Public Static void getElement (int [ ] a)
{
    int biggest = a[0];
    int Second.biggest = a[i];

    for (int i=0; i<a.length; i++)
    {
        if (a[i] > biggest)
        {
            Second biggest = biggest;
            biggest = a[i];
        }

        else if (a[i] > Second_biggest && a[i] != bigg
        {
            second biggest = a[i];
        }
    }
    Sop (biggest);
    Sop (second_biggest);
}

Page No.
Date : 9/4/24

→ Second Scenario

int [] a → [40, 40, 16, 24]
indexes: 0, 1, 2, 3

```
Public static void getElement(int[] a)
{
    int biggest = a[0];
    int secondbiggest = Integer.MIN_VALUE;

    for(int i=0; i<a.length; i++)
    {
        If(a[i] > biggest)
        {
            second_biggest = biggest;
            biggest = a[i];
        }
        else if(a[i] > second biggest && a[i] != biggest)
        {
            second_biggest = a[i];
        }
    }
    SOP(biggest);
    SOP(sec. biggest);
}
```

note:- Integer

SOP(Integer.MAX_VALUE);
2147483647

SOP(Integer.MIN_VALUE);
-2147483648

Page No.

Date : 9/1/24

Q(17) WAJP to Print Smallest and Second Smallest element of the array.

Page No.
Date : 9/1/21

Q[18] WAJP the shift all 0's to left and all 1's to right (without sorting).

I/p : [0,1,1,0,0,1,0,0]

O/p : [0,0,0,0,0,1,1,1]

```
                    int[]
Public static ~~void~~ Shif 01 (int[] a)
{
    int[] b = new int[a.length];
    int count1 = 0;
    for(int i=0; i<a.length; i++)
    {
        if(a[i] == 1)
            count1++;
    }
for(int i = a.length-1; count1 > 0; i--)
{
    b[i] = 1;
    count1--;
}
return b;
}
```

Page No.
Date : 20 / 1 / 24

H.W

Q.[19] For the given array of 0's, 1's and 2's Sort the elements (without sorting)

I/P:- [0, 2, 0, 1, 2, 1, 0, 2]

O/P: [0, 0, 0, 1, 1, 2, 2, 2]

```
                    int[]
Public Static void Shift Element(int[] a)
{   int count0 = 0, int count2 = 0; int n = a.length -1;
    int[] b = new int[a.length];
    for(int=0; i< a.length; i++)
    {
        if(a[i] == 0)
            count0++;
        else if(a[i] == 2)
            count2++;

        b[i] = 1;
    }
    for(int i=0; i<b.length; i++)
    {
        if(count0 > 0)
        {
            b[i] = 0; count0--;
        }
        if(count
        {
            b[n] = 2; count2--; n--;
        }
        if(count0 == 0 && count2 == 0)
            break;
    }
return b;
}
```

Page No.
Date : 10/1/24

~~(20) copy to~~

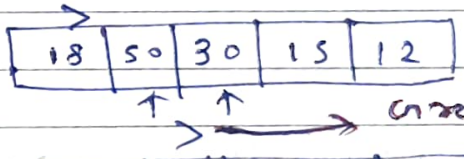
(Shorting)

↳ Shooting algoritham is use to short the ~~collection of~~ elements of array or collection of elements either in accending order or in decending order.

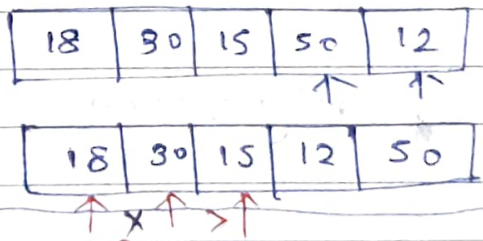
(1) Bubble Short!- Bubble short algorithm is use to combare two elements and swap the position of elements if required.

Algoritham :- a → | 50 | 18 | 30 | 15 | 12 |

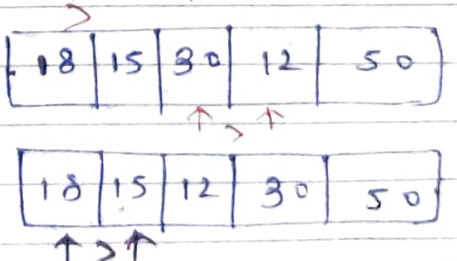
Step 1 -



Step 1 completed

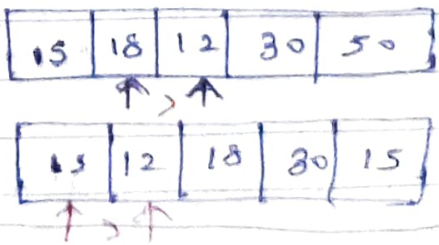


Step - 2



Step-2 completed

Step-3



Step-3 completed

Page No.
Date : 10/1/24

Step-1 | 12 | 15 | 18 | 30 | 50 |

Sorted Array

=> Algorithm a → | 18 | 12 | 10 | 30 | 40 |

Step-1 | 12 | 18 | 10 | 30 | 40 |

Step-1 completed | 12 | 10 | 18 | 30 | 40 |

Step-2 | 10 | 12 | 18 | 30 | 40 |

Shorted Array

=> Algorithm a → | 18 | 10 | 20 | 30 | 40 |

Step (1) | 10 | 18 | 20 | 30 | 40 |

=> Algorithm a → | 12 | 15 | 20 | 30 | 40 |

Shorted already Shorted

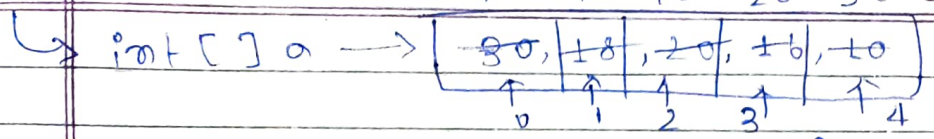
→ First Scenario
Loop => 4 times

→ Second Scenario
Loop => 2 times

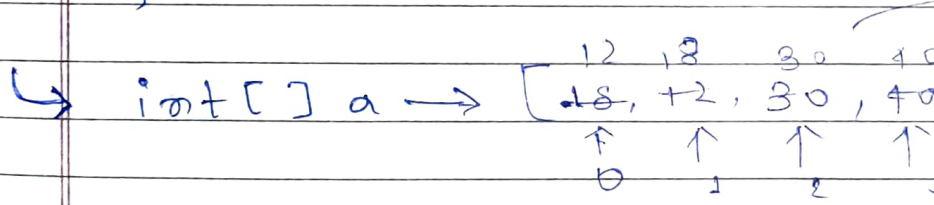
→ Third Scenario
Loop => 1 time

→ Fourth Scenario
Loop => 0 time

Page No.
Date : 11/01/24




```
Public static void sortArray(int[] a)
{
    int n = a.length - 1;
    for(int i > 0; i < n; i++)
    {
        for(int j = 0; j < n - i; j++)
        {   if(a[j] > a[j+1])
            {
                a[j] = a[j] + a[j+1];
                a[j+1] = a[j] - a[j+1];
                a[j] = a[j] - a[j+1];
            }
        }
    }
```




```
Public static void sortArray(int[] a)
{
    int n = a.length - 1;
    for(int i > 0; i < n; i++)
    {   int x = 0;
        for(int j = 0; j < n - i; j++)
        {   if(a[j] > a[j+1])
            {
                a[j] = a[j] + a[j+1];
                a[j+1] = a[j] - a[j+1];
                a[j] = a[j] - a[j+1];
                x++
            }
        }
        if(x == 0); break;
    }
}
```

(22)

```
Public static void sort(int[] a)
{
    int half = a.length/2;
for (int i=0, i< a.length; i++)
{
    for(int j=0; j<half-1; j++)
    {  if(a[j] > a[j+1])
       {
          a[j] = a[j] + a[j+1];
          a[j+1] = a[j] - a[j+1];
          a[j] = a[j] - a[j+1];
       }
for (int j = half; j< a.length-1; j++)
{       if(a[j] < a[j+1])
        {
          a[j] = a[j] + a[j+1];
          a[j+1] = a[j] - a[j+1];
          a[j] = a[j] - a[j+1];
        }
}
return a;
}
```

Page No.
Date :

(23)

```
Public static int[] sort(int[] a)
{
    int n = a.length-1;
    for(int i=0 i<n ,i++)
    {
        for(int j=0; j<n-1; j++)
        {
            if(a[j]>a[j+1])
            {
                a[j] = a[j] + a[j+1];
                a[j+1] = a[j] - a[j+1];
                a[j] = a[j] - a[j+1];
            }
        }
int ~~i~~ = ~~a.length-1~~;
    for(int i=0; i<a.length; i++)
    {
        if(~~i~~<a.length/2) sop(a[i]+" ");
        else
        {
            sop(a[~~i~~]+" ");
        } i--;
    }
}
```

Page No.
Date :

* Insertion Sort :-

↳ In insertion sort we compare the elements from right to left side. the right side element is considered is key this key is compare to all the left side element.

→ when an element in ~~right~~ left Side found bigger than key then we place the element one place in higher position.

→ If right side element is found smaller than key then key is inserted to the right of smaller element.

Page No.
Date :

0	1	2	3	4
~~17~~	~~18~~	~~30~~	~~30~~	~~15~~
15	17	~~30~~ 24 18	24	30

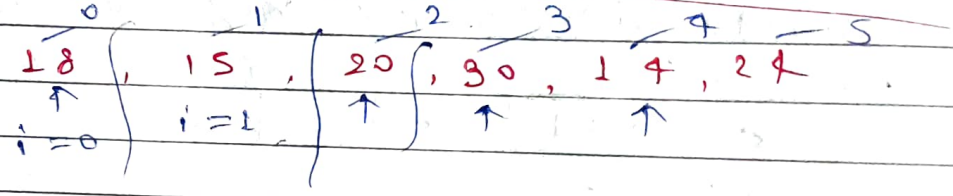
```
Public static void insertionSort(int[] a)
{
    for(int i=1; i<a.length; i++)
    {
        int key = a[i];
        int j = i-1;
        while(j>=0 && key<a[j])
        {
            a[j+1] = a[j];
            j--;
        }
        a[j+1] = key;
    }
}
```

Page No.
Date :

* Selection Sort :-

Selection Sort is used to select the smallest available element and put that element in left most position to achieve sorting

* Algorithm of Selection Sort




```
Public Static void selectionSort(int[] a)
{
 for(int i=0; i<a.length-1; i++)
 {
     int smallest = a[i];
     int min_index = i;
  for(int j = i+1; j<a.length; j++)
  {
     if(a[j]< smallest)
     {  smallest = a[j];
        min_index = j;
     }
  }
  a[min_index] = a[i];
  a[i] = smallest;
}
```

Page No.
Date : 16/1/24

18 16 20 10 30 14
↑

```
public static void SelectionSort(int[] a)
{
  for(int i=0; i<a.length-1; i++)
  {
    int min = a[i];
    int min_index = i;
    for(int j = i+1; j<a.length; j++)
    {
      if(a[j] < min)
      {
        min = a[j];
        min_index = j;
      }
    }
    a[min_index] = a[i];
    a[i] = min;
  }
}
```

Page No.
Date : 11/11/

* Searching :-

→ To search an element from an a
different algorithms that we can u
(i) Linear Search
(ii) Binary Search

(i) Linear Search :- In Linear Searchi
we search the element from the ver
first position one by one when the
target value is found then we stop
and return the expected result.

Q For a given array search weather
the given element is available in
an array or not by implementing lin
Search.

```
Public Static boolean Search(int [] a, i
                                target)
{
    for(int i = 0; i<a.length ; i++)
    {
        if(a[i] == = target)
        {
            return true;
        }
    }
return false;
}
```

Page No.
Date : 16/1/24

For return index

```
Public Static int Search(int[] a, int target)
{
  for(int i=0; i<a.length; i++)
  {
    if(a[i]==target)
    {
      return i;
    }
  }
return -1;
}
```

* Binary Search :- Binary Search is used to search an element from shorted array. In binary search every time array is divided into two section and we approach towards the target value.

↳ Algorithm for binary Search

Step-1 :- 12, 18, 30, 40, 60, 70

int min_index = 0;

int max_index = a.length;

int mid_index = $\frac{0+5}{2} = 2$

↳ First we have to store min_index and max index and we have to find out the value of mid index

* Step-2 :-

Now we have to cheakk weather the value at mid index is equals to target value or not. If it is equ then here only our searching wi complete and we can return the expected result.

- otherwise

(a) target value is less than mid va

target value < mid value

then min ~~value~~ index will remain sam but ~~the~~ max index will be one les than mid index

min-index = 0;

max-index = mid-index - 1;

(b) target value > mid-value

max-index = a.length - 1;

min-index = mid-index + 1;

→ Step-3

Same Process to be reabeated until the target is found

Page No.
Date : 17/1/24

[5, 8, 12, 24, 30, 40, 48]
low ↑ ↑ High

```
Public Static int binary Search (int[] a, int target)
{
    int low = 0;
    int max = a.length - 1;
    while (low <= max)
    {
        int mid = (low + max)/2;
        if (a[mid] == target)
            return mid;
        else if (a[mid] > target)
            max = mid - 1
        else
            low = mid + 1;
    }
    return -1;
}
public static void main (String[] args)
{
int[] a = {5, 18, 12, 24, 30, 40, 48};
    SOP ("Enter the Search");

    int target = Sc.nextInt();
int index = binary Search (a, target);
 if (index >= 0)
 {
   SOP (" element is available at index: " + index);
 }
 else
 {
   SOP (" Element is not available in array");
 }
}
```

Page No.
Date :

(by recursion)

```
          0   1   2   3   4   5   6
a → [ 5 , 8 , 12 , 24 , 30 , 40 , 48 ]
```

```
Public Static int binarySearch(int[] a, int
                                , int max, int
{
    int mid = (low + max) / 2;

    if(low > high max)
        return -1;
    if(a[mid] == target)
        return mid;
else if(a[mid] > target)
    return binarySearch(a, low, mid-1
                        target);
else
    return binarySearch(a; mid+1;
                        max, target);
}
Public Static void main(String[] args)
{
int[] a = { 5, 18, 12, 24, 30, 40, 48};
    SOP("Enter the nota Search");
    int target = Sc.next Int();

int index = binarySearch(a; 0; a.length, ta
if(index >= 0)
{ SOP(" Element is available at index :" +
}
```

Page No.
Date :

```
else
{
  System.out.println("Element is not available in array");
}
}
```

Page No.
Date :

32 WAJP to print the index and the value of the first non-repeating element in an array.

[15, 18, 30, 15, 20, 6, 18, 30, 12, 40]

```
Public static int getIndex(int[] a)
{
    int n = a.length;
    for(int i = 0; i<n; i++){
```

Page No.
Date :

33 WAJP to rotate all the elements of array 1 position to its right.

array[] = [1, 2, 3, 4, 5, 6, 7].

k = 2

Output : [6 7 1 2 3 4 5]

a → | 12 | 15 | 18 | 30 | 60 |
0 1 2 3 4

```
for(int j = 1; j <= n; j++)
{
  int x = a[a.length-1];
  for(int i = a.length - 2; i >= 0; i--)
  {
      a[i+1] = a[i];
  }
  a[i] = x;
}
```

→ For left shif

12, 15, 18, 30, 60

```
Public static void rotateleft(int [] a, int n)
for(int j = 1; j <= n, j++)
{
    int x = a[0];
for(int i = 0, i < a.length-1; i++)
{
    a[i] = a[i+1];
}
a[a.length-1] = x;
}
```

This is for single time