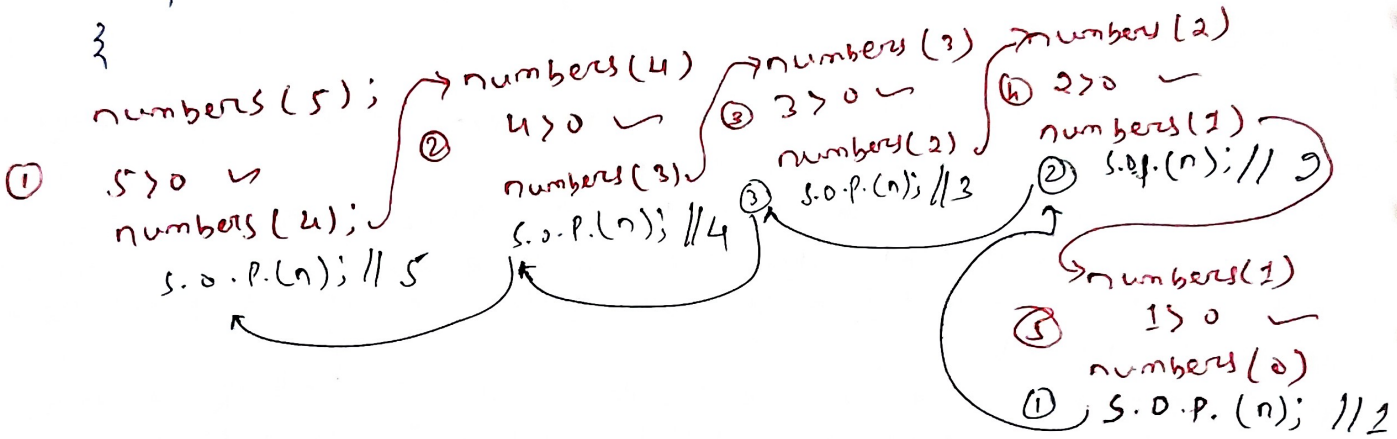
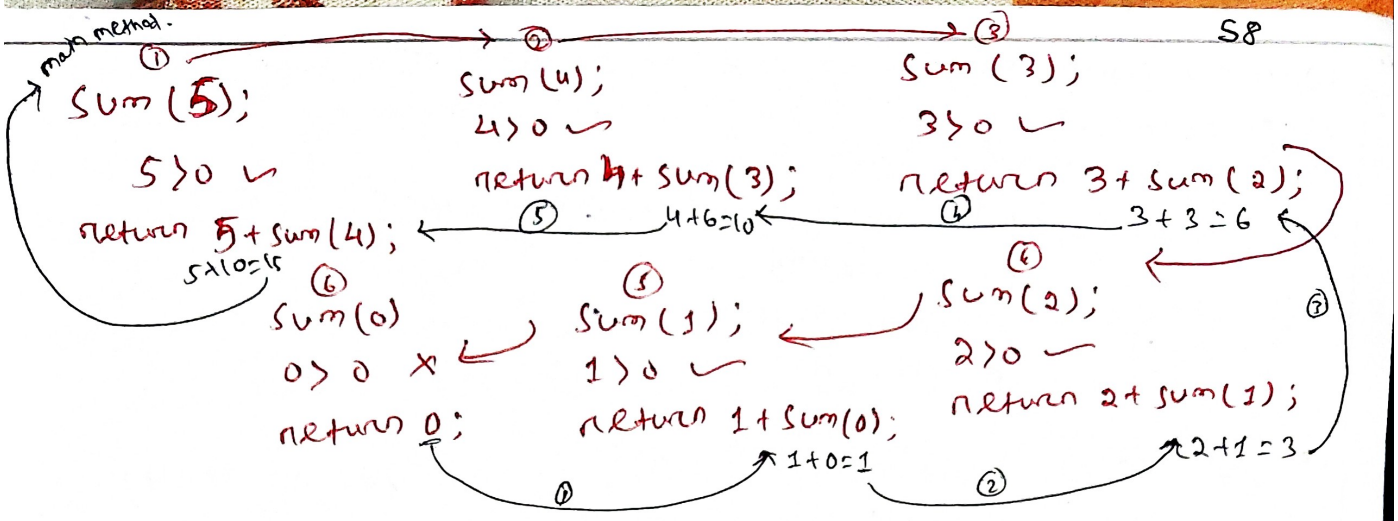
(2) Write a java program to print first n natural numbers using Recursion.

→
```
Class Natural
{
    Public static void main (String [] args)
    {
        int n = sc.nextInt();          let n=5;
        numbers(n);
    }
    Static void numbers (int n)
    {
        if (n>0)
        {
            numbers(n-1);
            System.out.print(n+" ");
        }
    }
}
```



(3) Write a java program to find sum of first n natural numbers using Recursion.

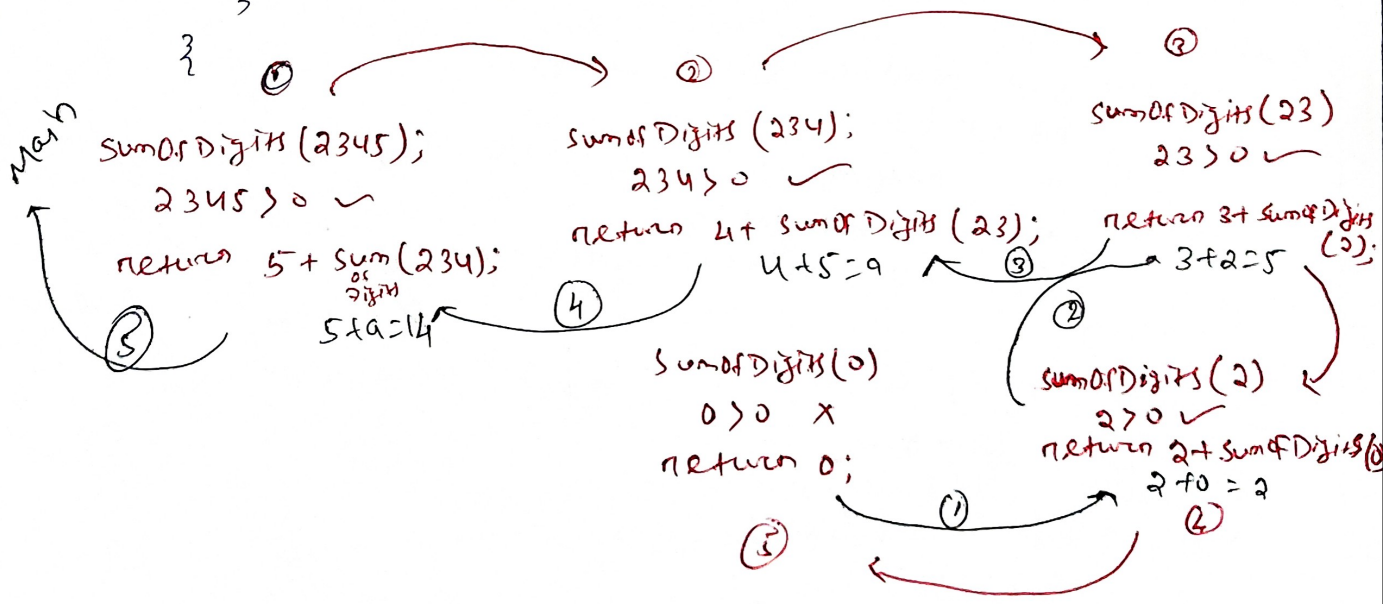
→
```
Class Sum{
    Public static void main (String [] args){
        int n, result;
        n = sc.nextInt();
        result = sum(n); // 15          let n = 5;
        System.out.println(result); // (15)
    }
    Static int sum (int n){
        if (n>0)
            return n + sum(n-1);
        else
            return n;
    }
}
```



) Write a Java program to find the sum of individual digits of a given number using recursion.

```
class sumofdigits {
    public static void main (String [] args) {
        int n, result;
        n = sc.nextInt;
        result = sumOfDigits (int n); // 14
        System.out.println (result); // 14
    }
    static int sumOfDigits (int n) {
        if (n>0)
            return ((n%10) + sumofDigits (n/10));
        else
            return n;
    }
}
```

Let n = 2345;



(5) Write a java program to find the reverse of a given number using recursion.

```
class Reverse {
    Public static void main (String [] args) {
        int n, result;
        n = sc.nextInt();
        reverse(n);
    }
/*  static void reverse(int n){
        if(n>0)
        { System.out.print(n%10);
          reverse(n/10);
        }
    }
*/
    Static int reverse(int n) {
        if (n>0)
        {
            System.out.print(n%10);
            return reverse(n/10);
        }
        else
            return n;
    }
}
```

Let n = 1234;

O/P = 4321

(6) Write a java program to find factorial of a given number using Recursion

```
class factorial{
    Public static void main (String [] args) {
        int n,result;
        n = sc.nextInt();
        result = factorial(n);
        System.out.println(result); // 120
    }
    Static int factorial (int n) {
        if (n==0)
            return 1;
        else
            return n * factorial(n-1);
    }
}
```

Let n = 5;

7) Write a java program to convert given binary number to a decimal number using recursion.

```
class BtD{
    public static void main (String [] args){
        int n, result;
        n=sc.nextInt();                          Let n=1111;
        result = btd(n);   // 15
        System.out.println(result); // 15
    }
    static int btd(int n){
        if (n==0)
          return 0;
        else
          return(n%10)+(2*btd(n/10));
        }
    }
```

8) Write a java program to convert given decimal number to a binary number using recursion.

```
class Dtb{
    public static void main (String [] args){
        int n, result;
        n=sc.nextInt();
        result = dtb(n);
        System.out.println(result);
    }
    static int dtb(int n){
        if (n==0)
          return 0;
        else
          return (n%2)+(10*dtb(n/2));
    }
}
```

16.11.23

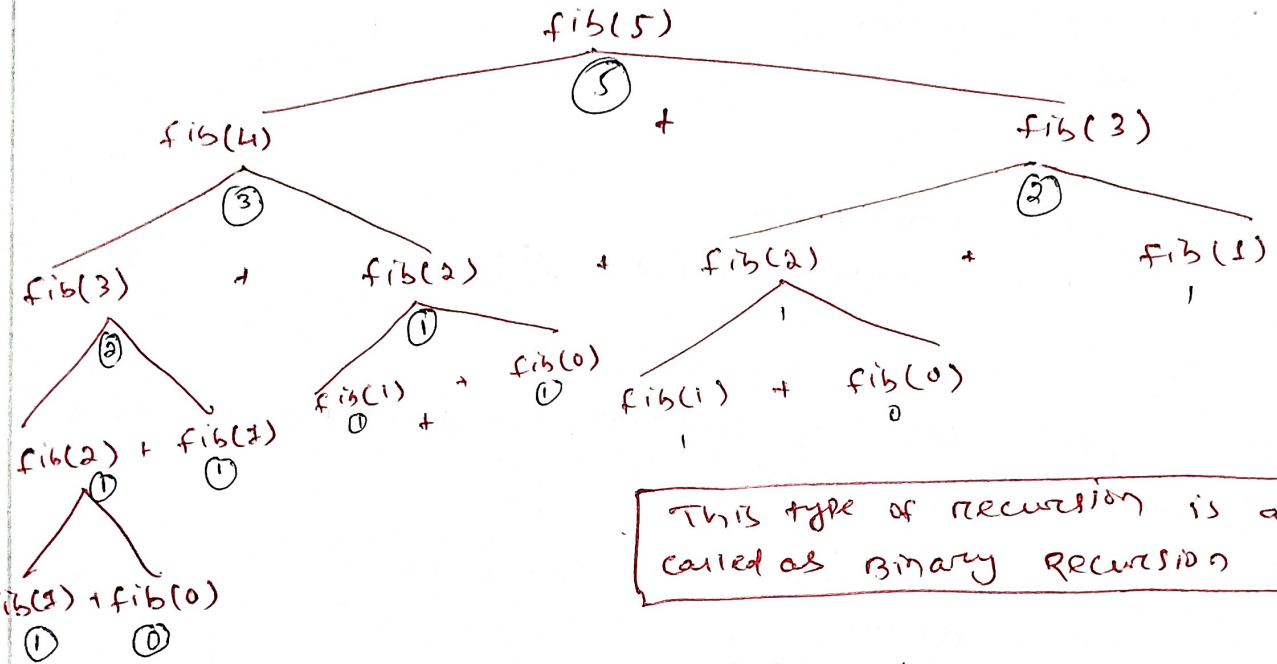
9) Write a java program to generate a fibonacci series upto a given number/limit using Recursion.

```
class fib{
    Public static void main (String [] args){
        for(i=0; i<limit; i++)
            S.o.p. (fib(i));
        }
```

[P.T.O.]

```
Static int fib(int n){
    if (n==0)
      return 0;
    else if (n==1)
      return 1;
    else
      return (fib(n-1)+fib(n-2));
  }
}
```

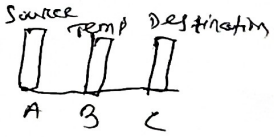
Let n=5



This type of recursion is also called as Binary Recursion

TOWER of HANOI USING RECURSION (GAME)

→ Basically it is a game in which we have to move the discs between three poles and rearrange the discs in ascending order such that no higher disc above the lower disc.



→ Move disc A to C → for 1 disc

→ Move disc from A to B using C
Move disc from A to C
Mode disc from B to C using A.
→ For 2-discs

Move (n-1) discs from A to B using C
Move disc from A to C.
Move (n-1) discs from B to C using A.
} for n-discs.

Code implementation

```
class TOH {
    public static void main (String [] args) {
        toh (3, 1, 2, 3);
    }
    void toh (int n, int A, int B, int C) {
        if (n>0) {
            toh (n-1, A, C, B);
            System.out.println("Move disc from" + A + "to" + C);
            toh (n-1, B, A, C);
        }
    }
```

(3 = No. of discs; 1 = A, 2 = B, 3 = C)

ARRAY

An array is a homogeneous collection of data elements that are stored under a unique name.

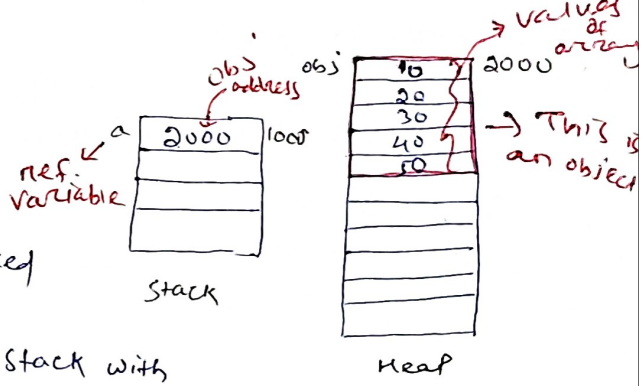
Declaration of an Array

Syntax : datatype arrayname [ ];

Ex: int a[ ]; char [ ]; float b[ ]; String s[ ]; etc.

Initialization of Array without value

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



- By the above syntax, an object created in heap memory and the address of the object will referred to stack (to a → 2000).
- The reference variable created in stack with an address (a → 1000)
- The values of the array stored in the heap with index value starting from 0.

Initialization of array with value

```
int a[ ] = new int[ ] {1, 2, 3, 4, 5};
```

For this type of initialization, we don't need to mention the size. According the input value, compiler will automatically gives size to the array. So if we mention size, then compiler will give an error.

```
int a[5] = {1, 2, 3, 4, 5};
```

For this type of initialization, we have to mention the size of array even if we are providing input values.

→ Array is of fixed size. We can't insert more values of the mentioned size.

Ex: int a[50];

We can insert only 50 integer records.

If we try to insert 51 records, then compiler will give an error : ArrayIndexOutOfBoundException.

→ Array support primitive type as well as non-primitive type (object).

→ Array has a default property as 'length' to get the size of an array.

NOTE

Arrays is a class in java which contains some pre-defined methods.

Classification of Array

→ Array can be classified as

(1) One dimension array

(2) Multi dimension array

- 2-D array
- 3-D array
- :
- n-D array.

One dimension array

→ One dimension array in Java is basically a linear array that allows its user to store multiple values of the same data type.

Array of objects

→ When we stores multiple objects in an array then it is called as Array of objects.

→ Ex:

```
class Student {
    int rollno;
    String name;
    Student (int rno, String nm) {
        rollno = rno;
        name = nm;
    }

class Arrayofobjects {
    Public static void main (String [] args) {
        Student a[] = new student [5];
        a[0] = new student (1, "sachin");
        a[1] = new Student (2, "kapil");
```

```
a[2] = new Student[3,"Sehwag"];
a[3] = new Student[4,"Dhoni"];
a[4] = new Student[5,"Rohit"];

for(int i=0; i<a.length; a++)
    S.o.pln(a[i].rollno+" "+a[i].name);
}
}
```

Array of primitive type

→ When we stores primitive type of values in an array then we go for primitive type array.

→ We can store only one type of primitive data in an array at one time.

① Write a java program to initialize an array containing 5 roll nos. of students and print the same on the screen.

→
```
int a[] = {101, 102, 103, 104, 105};
int i;
for(i=0; i<a.length; i++)
    S.o.pln(a[i]);
```

② Write a java program to store vowels in a character array and print them on the screen.

```
char a[] = {'A','E','I','O','U'};
int i;
for(i=0; i<a.length; i++)
    S.o.pln(a[i]);
```

③ Write a java program to store the salaries of 5 employeer in an organisation using array & print them on screen.

```
float a[] = {56456.45f, 21345565, 16987.89f, 21435.67f, 9840.78f};
int i;
for(i=0; i<a.length; i++)
    S.o.pln(a[i]);
```

④ Write a java program which stores 5 boolean values into an array and print the same on the screen.

```
boolean a[] = {true, false, true, false, true};
int i;
for(int i=0; i<a.length; i++)
    S.o.pln(a[i]);
```

(5) Write a java program to store the names of n-employees in an array and print them on the screen.

```
Scanner sc=new Scanner(System.in);
int i,n;
S.o.Pln("Enter size of array:");
n=sc.nextInt();
String a[]=new String[n];
for(i=0; i<a.length; i++)
{
    S.o.P.("Enter the name");
    a[i]=sc.next();
}
for(i=0; i<a.length; i++)
    S.o.Pln(a[i]);
```

17.11.23

(6) Write a java program to find the sum of all the elements of a given array.

→

```
int n,i,sum=0;
n=sc.nextInt();
int a[]=new int[n];
for(i=0; i<n; i++);
    a[i]=sc.nextInt();
for(i=0; i<n; i++)
    sum+=a[i];
S.o.Pln(sum);
```

(7) Write a java program to find the sum of ~~positive~~ even and ~~negative~~ odd numbers in a given array.

→

```
int n,i,esum=0,osum=0;
n=sc.nextInt();
int a[]=new int[n];
for(i=0; i<n; i++)
    a[i]=sc.nextInt();
for(i=0; i<n; i++)
{ if(a[i]%2==0)
    esum+=a[i];
  else
    osum+=a[i];
}
S.o.P.(esum);
S.o.P.(osum);
```

8) Write a java program to find the sum of positive and negative numbers in a given array.

```
int n, i, psum=0, nsum=0;
n= sc.nextInt();
int a[] = new int[n];
for (i=0; i<n ; i++)
    a[i] = sc.nextInt();
for (i=0; i<n ; i++)
{   if (a[i]>0)
        psum += a[i];
    else
        nsum += a[i];
}
S.o.P. (psum);
S.o.P. (nsum);
```

9) Write a java program to search an element in a given array along with position.

```
int i, i, number, position=0;
boolean search = false;
n= sc.nextInt();
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
number = sc.nextInt();
for (i=0; i<n; i++)
{   if (a[i] == number)
    {
        search = true;
        position = i;
        break;
    }
}
if (search)
    S.o.Pln (number + " is found at index : " + position);
else
    S.o.Pln (number + " is not present in the array");
```

(10) Write a java program which reads a source element and replace it with replacing element in a given array.

→
```
int n, i, number, replace;
n = sc.nextInt();
int a[] = new int[n];
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
number = sc.nextInt();
replace = sc.nextInt();
for (i=0; i<n; i++)
{ if(a[i]==number)
    a[i] = replace;
}
for (i=0; i<n; i++)
    S.o.p.(a[i] + " ");
```

(11) Write a java program to sort the elements in an array in ascending order.

→
```
int n,i, j,temp;
n = sc.nextInt();
int a[] = new int[n];
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
for (i=0; i<n; i++)
{
    for(j=i+1; j<a.length; j++)
    { if (a[i] > a[j])
        { temp = a[i]
          a[i] = a[j]
          a[j] = temp;
        }
    }
}
for (i=0; i<n; i++)
    S.o.p (a[i] + " ");
```

(12) Write a java program to sort the elements in an array in descending order.

```
int n, i, temp;
n = sc.nextInt();
int a[] = new int[n];
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
for (i=0; i<n; i++)
{ for (j= i+1; j<n; j++)
  { if (a[i] < a[j])
    { temp = a[i];
      a[i] = a[j];
      a[j] = temp;
    }
  }
}
for (i=0; i<n; i++)
    S.O.P. (a[i] + " ");
```

(13) Write a java program to find the largest element in a given array.

```
int n, i, largest;
n = sc.nextInt();
int a[] = new int [n];
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
largest = a[i];
for (i=1; i<n; i++)
{ if (a[i] > largest)
    largest = a[i];
}
S.O.P. (largest);
```

(14) Write a java program to find the smallest element in a given array.

```
int n, i, smallest;
n = sc.nextInt();
for int a[] = new int [n];
for (i= 0; i<n; i++)
    a[i] = sc.nextInt();
smallest = a[i];
```

```
for (i=0; i<n; i++)
{ if (a[i]< smallest)
        smallest = a[i];
}
S.O.P ( smallest);
```

(15) Write a java program to find the second largest element in a given array.

→
```
int n, i, largest, slargest;
n=sc.nextInt();
for (i=0; i<n ; i++ )
     a[i]=sc.nextInt();
largest = a[i];
slargest = 0;
for (i=1; i<n; i++)
{ if (a[i]>largest)
   { slargest = largest;
     largest = a[i];
   }
   else
   { if (a[i]>slargest)
        slargest = a[i];
   }
}
S.O.P. (slargest);
```

(or)

```
for (i=1; i<n; i++)
{ if (a[i]>largest)
     largest = a[i];
}
slargest=-999;
for (i=0;i<n; i++)
{ if (a[i]>slargest && a[i]!=largest)
     slargest = a[i];
}
S.O.P. (slargest);
```

(16) Write a java program to insert an element at desired location of a given array without deleting other elements.

→
```
int n, i, element, location;
n=sc.nextInt(); int a[] =new int [n+1];
for (i=0; i<n; i++)
     a[i] = sc.nextInt();
element = sc.nextInt();
location = sc.nextInt();
for (i=n; i>location; i--)
     a[i] = a[i-1];
a[i] = element;
n++;
for (i=0; i<n ; i++)
     S.O.P. (a[i]+" ");
```

18.11.23

(17) Write a java program to delete an element from any location of the array.

```
int n, i, location; n = sc.nextInt();
int a[] = new int[n];
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
location = sc.nextInt();
for (i=location; i<n; i++)
    a[i] = a[i+1];
n--;
for (i=0; i<n; i++)
    s.o.p(a[i]+" ");
```

(18) Write a java program which reads n-elements into two arrays and merge this two arrays into an resultant array.

```
int n, i, j=0;
n = sc.nextInt();
int a[] = new int[n];
int b[] = new int[n];
int c[] = new int[2*n];
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
for (i=0; i<n; i++)
    b[i] = sc.nextInt();
for (i=0; i<c.length; i++)
{   if (i<n)
        c[i] = a[i];
    else
    {   c[i] = b[j];
        j++;
    }
}
```

(or)

```
for (i=0; i<n; i++)
    c[i] = a[i];
j = i;
for (i=0; i<n; i++)
{   c[j] = b[i];
    j++;
}
```

(19) Write a java program which stores n-elements in an array & pick even elements from an array & place them into left side of the array & odd elements to the right side of the array.

```
int n, i, j, k, number;
n = sc.nextInt();
j = n-1; k = 0;
int a[] = new int[n];
for (i=0; i<n; i++) {
    number = sc.nextInt();
    if (number%2 == 0)
    {   a[k] = number;
        k++; }
    else
    {   a[j] = number;
        j--;
    }
}
```

```
int n, i, j, temp;
int a[] = new int[] {1,2,3,4,5,6};
n = a.length;
for (i=0; i<n; i++)
{
    j = n-1;
    if (a[i]%2 != 0)
    {
        temp = a[i];
        while (j>i)
        {
            if (a[j]%2==0)
            {
                a[i] = a[j];
                a[j] = temp;
                break;
            }
            j--;
        }
    }
}
```

(or)

```
int n, i, min, max;
n = sc.nextInt();
a[] = new int[n];
for(i=0; i<n; i++)
    a[i] = sc.nextInt();
min = 0;
max = n-1;
while (min < max)
{ while (a[min]%2==0 && min<max)
      min++;
  while(a[max]%2==1 && min<max)
      max--;
  if (min<max)
  {
    temp = a[min];
    a[min] = a[max];
    a[max] = temp;
  }
  min++;
  max--;
}
```

(20) Write a java program to perform left rotation of an array.

```
int i, j, n, temp, nr;
int a[] = new int[] {1,2,3,4,5};
n = a.length;
nr = sc.nextInt();
for(i=0; i<nr; i++)
{
    temp = a[0];
    for (j=0; j<n-1; j++)
        a[j] = a[j+1];
    a[n-1] = temp;
}
```

nr → Number of Rotation

(21) Write a java program to perform right rotation of an array.

```
int i, j, n, temp, nr;
int a[] = new int[] {1,2,3,4,5};
n = a.length;
nr = sc.nextInt();
for(i=0; i<nr; i++)
{   temp = a[n-1];
    for (j=n-1; j>0; j--)
        a[j] = a[j-1];
    a[0] = temp;
}
```

(21) Write a java program to find the maximum occurring element in a given array or most frequently occurred elements in a given array.

```
int i, j, n, count, count1=0, element=0;
int a[] = new int[] {2,2,3,2,3,4,2,5,6,2,7,2,9,13};
n=a.length;
for (i=0; i<n; i++)
{ count = 1;
  for (j=i+1; j<n; j++)
  {  if (a[i]==a[j])
        count++;
  }
  if (count > count1)
  {  count1 = count;
     element = a[i];
  }
}
```

(22) Write a java program to find second repeated element in a given array. (if i/p array = [4,2,5,7,11,14,5,13] then o/p = 5)

```
int n, i, j, fre, sre;
int a[] = new int[] {4,2,5,7,11,14,5,13};
n= a.length;
fre = 0;
sre = 0;
for (i=0; i<n; i++)
{ for (j=i+1; j<n; j++)
  {  if (a[i]==a[j])
     {  if (fre==0) {
           fre = a[i]; break; }
        else if (a[i] != fre)
        {  sre = a[i];
           break;
        }
     }
  }
  if (sre != 0)
     break;
}
if (sre != 0)
   s.o.p.(sre);
else
   s.o.p.("second repeated element is not in the array");
```

(23) Write a java program to remove the duplicate elements in a given array.

```
int i, j, k, n;
int a[] = new int[]{45, 23, 12, 45, 33, 23, 67, 11, 72, 11};
n = a.length;
for (i = 0; i < n; i++)
{
    for (j = i+1; j < n; j++)
    {
        if (a[i] == a[j])
        {
            for (k = j; k < n-1; k++)
                a[k] = a[k+1];
            n--;
            j--;
        }
    }
}
```

(24) Write a java program to find out the missing element in the array.

```
int i, j, n, x;
int a[] = new int[]{2, 8, 11, 17, 23};
n = a.length;
x = a[0];
for (i=0; i<n; i++)
{
    if (a[i]-i != x)
    {
        while (x < a[i]-i)
        {
            s.o.p.((x+i)+" ");
            x++;
        }
    }
}
```

21.11.23

(25) Write a java program to print the leader element from an array. (Leader element is the element which is greater than its next coming element.)

```
int a[] = new int[]{14, 17, 3, 4, 5, 2};
int i, j, n;
n = a.length;
for (i = 0; i < n; i++)
{
```

```
for (j=i+1; j<n; j++)
{
    if (a[i] < a[j])
        break;
}
if (j==n)
    S.o.p. (a[i] + " ");
}
```

(26) Write a java program to print the prime numbers from an array.

→
```
int a[] = {1, 35, 34, 65, 55, 23, 2, 5, 3, 21};
int i, j; boolean flag;
for (i=0; i<a.length; i++)
{   flag = false;
    if (a[i] == 1)
        flag = true;
    for (j=2; j<=a[i]/2; j++)
    {
        if (a[i] % j == 0)
        { flag = true;
          break;
        }
    }
    if (!flag)
        S.o.p. (a[i] + " ");
}
```

(27) Write a java program to print the palindrome numbers from an array.

→
```
int a[] = {151, 121, 142, 243, 567, 272};
int i, temp, rev;
for (i=0; i<a.length; i++)
{ rev=0;
  temp = a[i];
  while (temp>0)
  { rev = rev*10 + (temp%10);
    temp = temp/10;
  }
  if (rev == a[i])
      S.o.p. (a[i] + " ");
}
```

(28) Write a java program to print Armstrong numbers from an array

→
```
int a[] = {153, 1, 2, 3, 142, 370, 407, 272};
int i, temp, count, sum, rem;
for (i=0; i<a.length; i++)
{ count = 0;
  sum = 0;
  temp = a[i];
  while (temp>0)
  { temp = temp/10;
    count ++;
  }
  temp = a[i];
  while (temp>0)
  { rem = temp%10;
    sum = sum + (int)(Math.POW(rem, count));
    temp = temp/10;
  }
  if (sum == a[i])
    S.O.P. (a[i] + " ");
}
```

(29) Write a java program to print ~~armstrong~~ strong numbers from an array.

→
```
int a[] = {145, 1, 2, 3, 142, 370, 407, 272};
int i, temp, count, sum, rem, fact;
for (i=0; i<a.length; i++)
{
  count = 0;
  sum = 0;
  temp = a[i];
  while (temp>0)
  { temp = temp/10;
    count ++;
  }
  temp = a[i];
  while (temp>0)
  { rem = temp%10;
    fact = 1;
    while (rem>0)
    { fact = fact * rem;
      rem--;
    }
    sum = sum + fact;
    temp = temp/10;
  }
  if (sum == a[i])
    S.O.P. (a[i] + " ");
}
```

Write a java program to print the ~~STRONG~~ NUMBERS from an array.

```
int a[] = {1,2,6,7,11,12,101,105,103,497,496};
int i, temp, j, rev, sqr1, sqr2;
for(i=0; i<a.length; i++)
{
    sqr1 = (a[i]*a[i]);
    temp = a[i];
    rev = 0;
    while (temp>0)
    { rev = rev*10 + (temp%10);
      temp = temp/10;
    }
    rev = rev*rev;
    sqr2 = 0;
    while (rev>0)
    { sqr2 = sqr2*10 + (rev%10);
      rev = rev/10;
    }
    if (sqr2==sqr1)
        S.O.P. (a[i]+" ");
}
```

Write a java program to print the PERFECT NUMBERS from an array.

```
int a[] = {6,96,28,493,147,272,497,496};
int i,j, temp, sum;
for (i=0; i<a.length; i++)
{   sum=0;
    for (j=1; j<a[i]; j++)
    {
        if (a[i]%j==0)
            sum = sum+j;
    }
    if (sum==a[i])
        S.O.P. (a[i]+" ");
}
```

Write a java program to print fibonacci terms present in an array.

```
int a[] = {1,35,34,65,55,23,2,5,3,21};
int fib[] = {0,1,1,2,3,5,8,13,21,34,55};
int i,j;
for(i=0; i<a.length; i++)
{   for(j=0; i<fib.length; j++)
    {
```

```
        if (a[i] == 1)
        {   s.o.p. (a[i] + " ");
            break;
        }
        else if (a[i] == fib[j])
        {   s.o.p. (a[i] + " ");
            break;
        }
    }
}
```

TWO-DIMENSIONAL ARRAY

→ Two dimensional array can befined in simple words as array of arrays. Data in 2-d array stored in tabular form.

→ 2-D array represented with [ ] [ ].

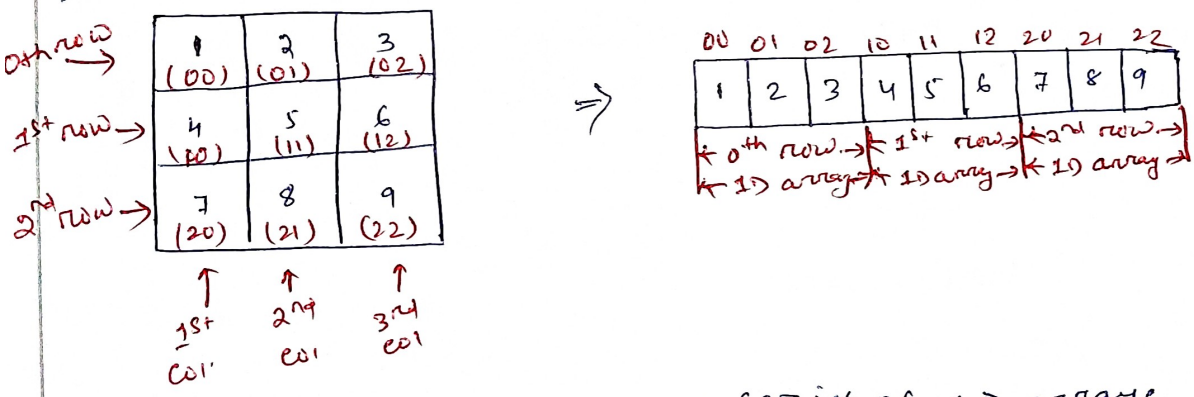
ROWSIZE ↑ ↑ COLUMNSIZE

Ex: [3] [3]

3 x 3 array which can stores 3*3=9 values.

→ 2-D arrays stores values in series of 1-D arrays.

Ex:



→ We can call a 2-D array as series of 1-D arrays.

Declaration of 2-D Array:

Syntax: datatype arrayname [rowsize] [colsize];

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

double a[][] = new double [4][4];

Initialization of 2-D Array

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

Rowsize = 3
Col size = 3

We can also also initialize as different 1-D array size as below.

```
int a[] = new int[][]{{1,2,3},{4,5},{7}};
```

Here row size = 3.
1st 1-D array size = 3
2nd 1-D array size = 2
3rd 1-D array size = 1.

→ (size of the 2-D array = 3×3)

for-each loop

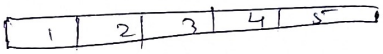
for-each loop is also known as enhanced for loop.
It starts with the keyword for like a normal for loop.
Instead of declaring and initializing a loop counter variable, we declare a variable that is the same type of the base type of the array, followed by a colon (:) which is then followed by the array name.
In the loop body, we can use the loop variables that we created rather than using an indexed array element.

Syntax:

```
for (datatype variablename : arrayname)
{
    Statements using variablename;
}
```

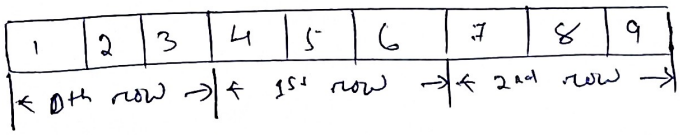
NOTE:
datatype must be same as the array datatype

Ex:
For 1-D Array




```
int a[] = {1, 2, 3, 4, 5};
for (int element : a)
{
    S.o.p. (element + " ");
}
```

For 2-D Array




```
for (int [] i : a)
{
    for (int j : i)
        S.o.p. (j + " ");
    S.o.Pln();
}
```

using nested for loop for 2-D array

```
for(i=0; i<a.length; i++)
{ for (j=0; j<a[i].length; j++)
        S.o.p. (a[i][j]);
    S.o.Pln();
}
```

① Write a java program which stores n-elements into a 2-D array and print the same in the form of a matrix.

```
int i, j, row, col;
int a[][] = new int[10][10];
row = sc.nextInt();
col = sc.nextInt();
for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
        a[i][j] = sc.nextInt();
}
for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
        S.o.p.(a[i][j] + " ");
    S.o.p.ln();
}
```

② Write a java program to find the sum and product of all elements of a 2-D array.

→

```
int a[][] = new int[10][10];
row = sc.nextInt();
col = sc.nextInt();
for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
        a[i][j] = sc.nextInt();
}
sum = 0;
prod = 1;
for (i=0; i<row; i++)
{
    for (j=0; j<col; j++)
    {
        sum = sum + a[i][j];
        prod = prod * a[i][j];
    }
}
S.o.p. (sum);
S.o.p. (prod);
```

3) Write a java program to find the sum of principle diagonal and other diagonal element of the 2-D array.

```
int i, j, row, col, psum, osum;
int a[][] = new int[10][10];
row = sc.nextInt();
col = sc.nextInt();
for(i=0; i<row; i++)
{ for(j=0; j<col; j++)
      a[i][j] = sc.nextInt();
}
psum = 0;
osum = 0;
for(i=0; i<row; i++)
{ for(j=0; j<col; j++)
  { if(i==j)
      psum = psum + a[i][j];
  // if(i+j==col-1)
    if(a[i][j]==a[i][col-i-1])
      osum = osum + a[i][j];
  }
  s.o.println();
}
s.o.println(psum);
s.o.println(osum);
```

4) Write a java program to find the largest number along with it position in a 2-D array.

```
for(i=0; i<row; i++)
{ for(j=0; j<col; j++)
      a[i][j] = sc.nextInt();
}
largest = a[0][0];
p1 = 0;
p2 = 0;
for(i=0; i<row; i++)
{ for(j=0; j<col; j++)
  { if(a[i][j] > largest)
    { largest = a[i][j];
      p1 = i;
      p2 = j;
    }
  }
}
s.o.p(largest + "found at " + p1 + ", " + p2);
```

(5) Write a java program to print upper and lower diagonal element of a given matrix.

if i/p : 1 2 3
4 5 6
7 8 9

⇒ o/p : 2 3
4 6
7 8

→
```
for(i=0; i<row; i++)
{
  for (j=0; j<col; j++)
  {
    // if (i!=j)
    //       S.o.p.(a[i][j]+" ");
    if (i<j || i>j)
        S.o.p.(a[i][j]+" ");
    else
        S.o.p(" ");
  }
  S.o.println();
}
```

(6) Write a java program to check whether the given matrix is a diagonal matrix or not. (Diagonal matrix means only non-zero elements of a diagonal from top left to down right & other elements are zero).

Ex:

1 0 0
0 2 0
0 0 3
↑ Diagonal matrix

1 1 0
0 2 0
1 0 2
↑ Not a diagonal matrix

0 0 1
0 2 0
3 0 0
↑ Not a diagonal matrix

→
```
for (i=0; i<row; i++)
{
  for (j=0; j<col; j++)
  {
    if (i!=j)
    {
      if (a[i][j] != 0)
      {
        flag = false;
        break;
      }
    }
    else
    {
      if (a[i][j] == 0)
      {
        flag = false;
        break;
      }
    }
  }
```

```
if (!flag)
    break;
}
if (flag)
    S.o.p. ("Diagonal Matrix");
else
    S.o.p. ("Non-diagonal matrix");
```

23.11.23

Write a java program to find sum of each row and each column of a given 2-D Array. Let Array be $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

```
for (i=0; i<row; i++)
{ for (j=0; j<col; j++)
      a[i][j] = sc.nextInt();
}

for (i=0; i<row; i++)
{
   rsum=0;
   csum=0;
/*  for (j=0; j<col; j++)
    {
       rsum += a[i][j];
       csum += a[j][i];
    }
    S.o.Pln("sum of row"+i+" is :"+rsum);
    S.o.Pln("sum of column"+i+" is :"+csum);
}
*/
                    (or)
   for (i=0; i<row; i++)
   {
      rsum=0;
      for (j=0; j<col; j++)
         rsum += a[i][j];
      S.o.Pln("sum of row"+i+" is :"+rsum);
   }
   for (i=0; i<row; i++)
   {
      csum=0;
      for (j=0; j<col; j++)
         csum += a[i][j];
      S.o.pln ("sum of column"+j+" is :"+csum);
   }
```

O/P

```
Sum of row 1 = 6
Sum of column 1 = 12
Sum of row 2 = 15
Sum of column 2 = 15
Sum of row 3 = 24
Sum of column 3 = 18
```

O/P

```
Sum of row 1 = 6
Sum of row 2 = 15
Sum of row 3 = 24
Sum of column 1 = 12
Sum of column 2 = 15
Sum of column 3 = 18
```

(8) Write a java program to find the maximum rowsum of a given 2-D array.

→
```
maxrow = 0;
for (i=0; i<row; i++)
{
  for (j=0; j<col; j++)
      rsum += a[i][j];
  if (rsum > maxrow)
      maxrow = rsum;
}
S.o.p (maxrow);
```

(9) Write a java program to exchange any two rows of a given matrix/2-D array.

→
```
S.o.p("Enter rows you want to exchange:");
n1 = sc.nextInt();
n2 = sc.nextInt();
for(i=0; i<row; i++)
{
  temp = a[n1-1][i];
  a[n1-1][i] = a[n2-1][i];
  a[n2-1][i] = temp;
}
```

(10) Write a java program to exchange any two columns of a given matrix/2-D array.

```
S.o.p("Enter columns you want to exchange:");
n1 = sc.nextInt();
n2 = sc.nextInt();
for (j=0; j<col; j++)
{
  temp = a[j][n1-1];
  a[j][n1-1] = a[j][n2-1];
  a[j][n2-1] = temp;
}
```

27.11.23

(11) Write a java program to sort each row of a given 2-D array/matrix.

→
```
for (i=0; i<row; i++){
  for(j=0; j<col; j++){
    for (k=j+1; k<col; k++){
      if (a[i][j] > a[i][k]){
        temp = a[i][j];
        a[i][j] = a[i][k];
        a[i][k] = temp;  }}}}
```

12) Write a java program to sort each column of a given 2-D array/matrix.

```
for (j=0; j<col; j++) {
    for(i=0; i<row; i++) {
        for (k=i+1; k<row; k++) {
            if (a[i][j]>a[k][j]) {
                temp = a[i][j];
                a[i][j] = a[k][j];
                a[k][j] = temp;
            }
        }
    }
}
```

13) Write a java program to check whether the given matrix is a sparse matrix or not. (The matrix contains more '0' than other elements is called a sparse matrix).

```
size = row * col;
for(i=0; i<row; i++){
    for (j=0; j<col; j++) {
        if (a[i][j]==0)
            count++;
    }
    if(count > size/2)
        break;
}
if (count > size/2)
    S.o.println("The given matrix is a sparse matrix");
else
    S.o.println("The given matrix is not a sparse matrix");
```

14) Write a java program to find multiplication of two matrix and store the result in another matrix.

```
S.o.println("Enter the order of matrix a");
r1 = sc.nextInt();
c1 = sc.nextInt(); int a[][] = new int[r1][c1];
s.o.println("Enter the order of matrix b");
r2 = sc.nextInt();
c2 = sc.nextInt();
int b[][] = new int[r2][c2];
int c[][] = new int[r1][c2];
if (c1==r2) {
 s.o.println("Multiplication is possible");
 for(i=0; i<r1; i++) {
     for(j=0; j<c2; j++){
         c[i][j] = 0;
         for(k=0; k<r2; k++)
             c[i][j] = c[i][j] + a[i][k] * b[k][j];
     }
 }
```

(15) Write a java program to find addition of two matrix.

```
int row, col, i, j;
int a[][] = new int [row][col];
int b[][] = new int [row][col];
int c[][] = new int [row][col];
for (i=0; i<row; i++) {
    for (j=0; j<col; j++) {
        c[i][j] = a[i][j] + b[i][j];
    }
```

(16) Write a java program to find subtraction of two matrix.

```
for (i=0; i<row; i++) {
    for (j=0; j<col; j++)
        c[i][j] = a[i][j] - b[i][j];
    }
```

Searching Techniques

28.11.23

→ Searching means we have to find an element in an array.

→ We can achieve the searching by
(1) Linear Search / Sequential Search
(2) Binary Search.

Linear search

→ In this search technique, each element of the array will check until the desired or required element found

→ It is suitable for small-sized array.

→ Best case of Linear search: If the required element is at 1st position of the array.

Average case of Linear search: If the required element is at n/2 position of the array.

Worst case of Linear search: If the required element is at n position of the array.

Ex:

10	15	20	25	30	35	40	45	50
0	1	2	3	4	5	6	7	8

n = 9;

Best case: If required element = 10. (at 0th index)

Average case: If required element = 30 (at 4th index)

Worst case: If required element = 50 (at 8th index).

→ To avoid worst case or to save time, we have to go for binary search

Binary Search

The prerequisite of binary search is the element of array must be in sorted order.

It repeatedly compares the middle element of the target element, and narrows down the search range by half based on the comparison result.

It is efficient for large sorted arrays.

The algorithm be

```
low = 0; high = n-1; KE = sc.nextInt();
mid = (low + high)/2;
if (KE == a[mid]) {
    KE is found;
    break;
  }
else if (KE > a[mid])
      low = mid + 1;
else
    high = mid - 1;
```

We have to rotate the loop until we get the KE.

(1) Write a java program to find/search an element from an array using binary search.

```
main(){
S.O.Pln("Enter the element"); int a[] = new int [] {12, 25, 17, 6, 102, 77};
ke = sc.nextInt(); n = a.length;
sort(a, n);
binarySearch(a, n, ke);
}
static void sort(int a[], int n)
{
  int i, j, temp;
  for(i=0; i<n; i++) {
    for(j=i+1; j<n; j++) {
        if(a[i] > a[j])
        { temp = a[i];
          a[i] = a[j];
          a[j] = temp;
        }
    }
  }
}
static void binarySearch(int a[], int n, int ke) {
      int low, high, mid, i, found=0;
      low = 0;
      high = n-1;
  for(i=0; i<n; i++) {
      mid = (low + high)/2;
```

```
if (ke == a[mid]) {
    found = 1;
    break;
}
else if (ke > a[mid])
    low = mid+1;
else
    high = mid-1;
}
if (found == 1)
    S.o.pln (ke+" is present in array");
else
    S.o.pln (ke+" is not present in array");
```

Sorting Techniques

→ Sorting is nothing but arranging the array elements either in ascending or descending order.

Bubble Sort

→ It will rotate for n-1 passes.

→ For each pass, largest element is placed at right most side of an array.

① Write a java program to sort an array using bubble sort.

→
```
int i, n;
main() {
int a[] = new int [] {12, 25, 17, 6, 102, 77};
n = a.length;
bubbleSort(a, n);
}
static void bubbleSort (int a[], int n) {
    int i, j, temp, p;
    for (i=0; i<n-1; i++) {
        for (j=0; j<n-1-i; j++) {
            if (a[j] > a[j+1]) {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
        for (p=0; p<n; p++)
            S.o.p (a[p]+" ");
        S.o.pln ( );
    }
}
```

Insertion Sort

1.12.23

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.

The first element in the array is assumed to be sorted.

Take the second element and compare it with first element. If it is less than first element then it will place left to the element otherwise it will print right to the element.

Take the third element and compare it with the elements left to it. If the element is less than other two elements then it will place beginning of the array otherwise it will place accordily to its position by comparing with other two values.

We have to continue this process until all elements in the array got checked.

Ex:

5 12 87 25 9 65 98 34

5 12 87 25 9 65 98 34

5 12 87 25 9 65 98 34

5 12 87 25 9 65 98 34 ⇒ 5 12 25 87 9 65 98 34

5 12 25 87 9 65 98 34 ⇒ 5 9 12 25 87 65 98 34

5 9 12 25 87 65 98 34 ⇒ 5 9 12 25 65 87 98 34

5 9 12 25 65 87 98 34

5 9 12 25 65 87 98 34 ⇒ 5 9 12 25 34 65 87 98

o/p ⇒ 5 9 12 25 34 65 87 98

code

```
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;
}
```

Selection Sort

Selection sort is an in-place comparison-based sorting algorithm that is known for its simplicity over other sorting algorithms.

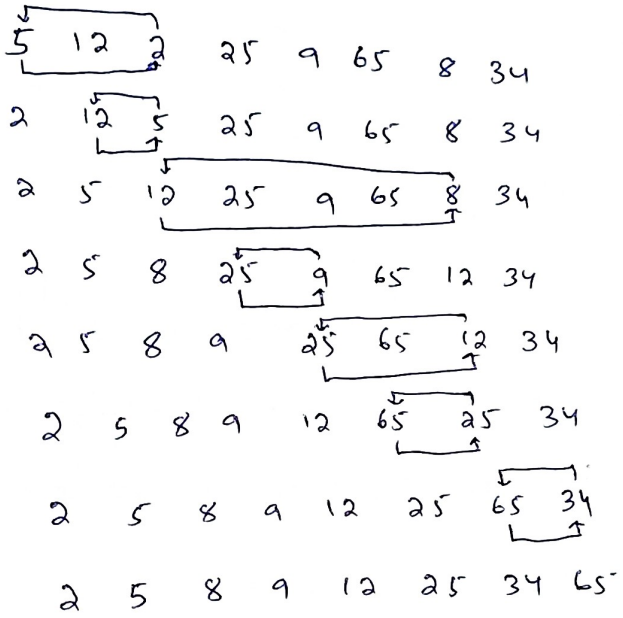
The selection sort technique involves choosing the array's smallest element and swapping it with array's first element (if sorting in ascending order).

The next step is to swap the second smallest member in the array with the element and so on.

Continue this process until the last element of array got sorted.

This sorting technique is the least used sorting technique in application development.

Ex:



Code:

```
for (i=0; i<n-1; i++) {
    min = a[i];
    pos = i;
    for (j=i+1; j<n; j++) {
        if (a[j] < min) {
            min = a[j];
            pos = j;
        }
    }
    a[pos] = a[i];
    a[i] = min;
}
```

Quicksort

Quicksort is the fastest and one of the most efficient sorting algorithms technique, which follows divide and conquer algorithm.

It picks an element as pivot and partitions the given array around the picked pivot.

The pivot element can be first element or last element or any random element of array, but most frequently first & last elements are pivot element.

If we select the first element as pivot element, then make second element as 'p' and last element as 'q'.

P search for larger element than pivot and q search for smaller element than pivot.

Where p got larger element and q got smaller element, then we have to swap p and q.

When p meets q at same element or q crosses p, then we have to swap pivot element with q.

Now pivot element will be at its required position such that smaller element are in the left side and larger element are in the right side of pivot element.

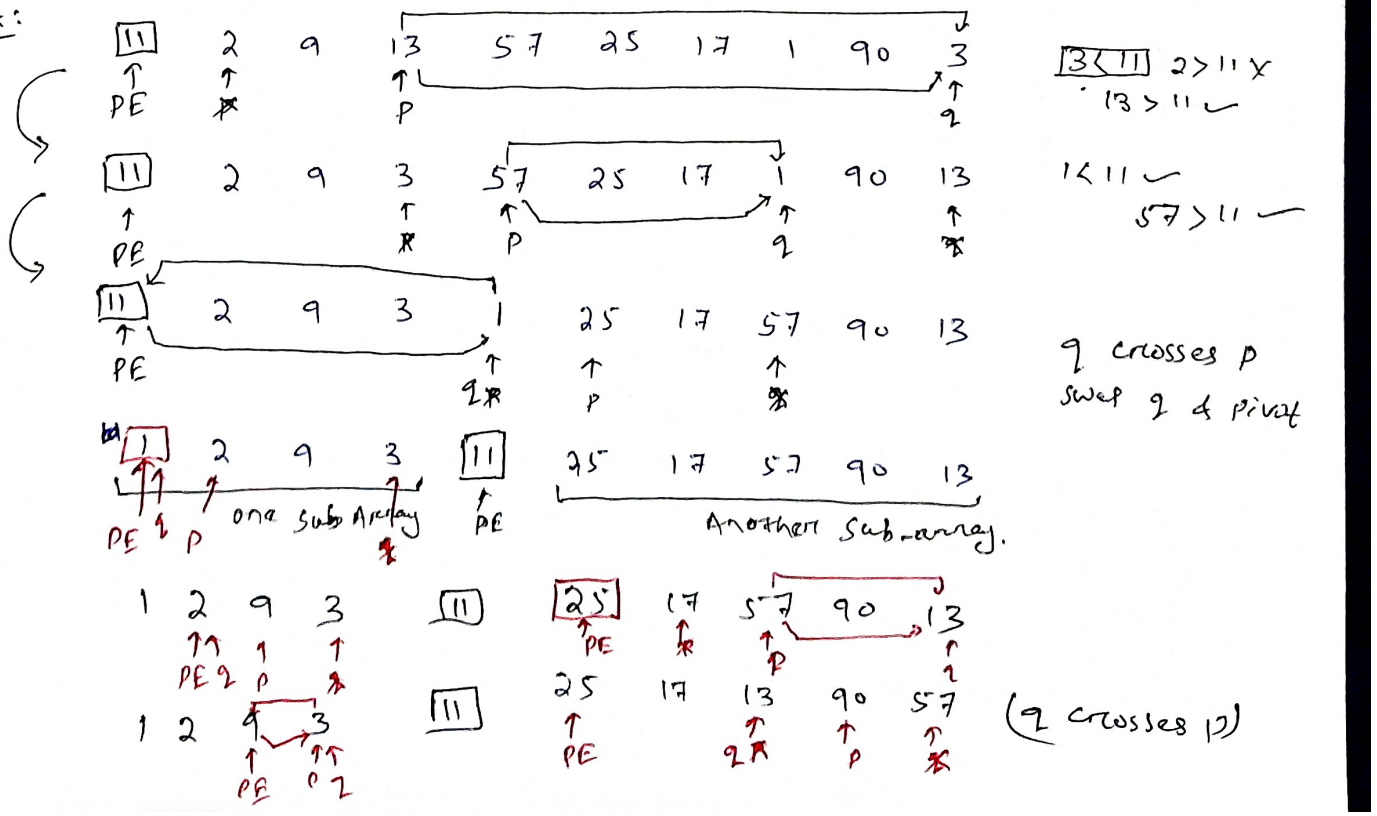
Now we have to take left side array elements as one sub array and right side array element as another sub array.

We have to continue the above steps until the sub array get sorted.

When p & q are at same element, swap p with pivot element.

When q & pivot are at same element, then the pivot element is at right position. No need to swap.

Ex:



Code of QuickSort

```
class QuickSort {
    public static void main(String[] args) {
        int n;
        int a[]= {11,2,9,13,57,25,17,1,90,3};
        n=a.length;
        quickSort(a,0,n-1);
        for (int e:a)
            System.out.print(e+" ");
    }
    static void quickSort(int a[], int lower, int upper) {
        int i;
        if (upper>lower)
        {
            i=split(a,lower,upper);
            quickSort(a,lower,i-1);
            quickSort(a,i+1,upper);
        }
    }
    static int split(int a[], int lower, int upper) {
        int i,p,q,t;
        p=lower+1;
        q=upper;
        i=a[lower];
        while (q>=p) {
            while (p<=q && a[p]<=i)
                p++;
            while (a[q]>i)
                q--;
            if (q>p) {
                t=a[p];
                a[p]=a[q];
                a[q]=t;
            }
        }
        t=a[lower];
        a[lower]=a[q];
        a[q]=t;
        return q;
    }
}
```

Merge Sort

→ Merge sort is a well-known and successful sorting approach that uses the divide and conquer algorithm to divide a problem into many sub-problems.

→ It is one of the most popular and efficient sorting algorithm.

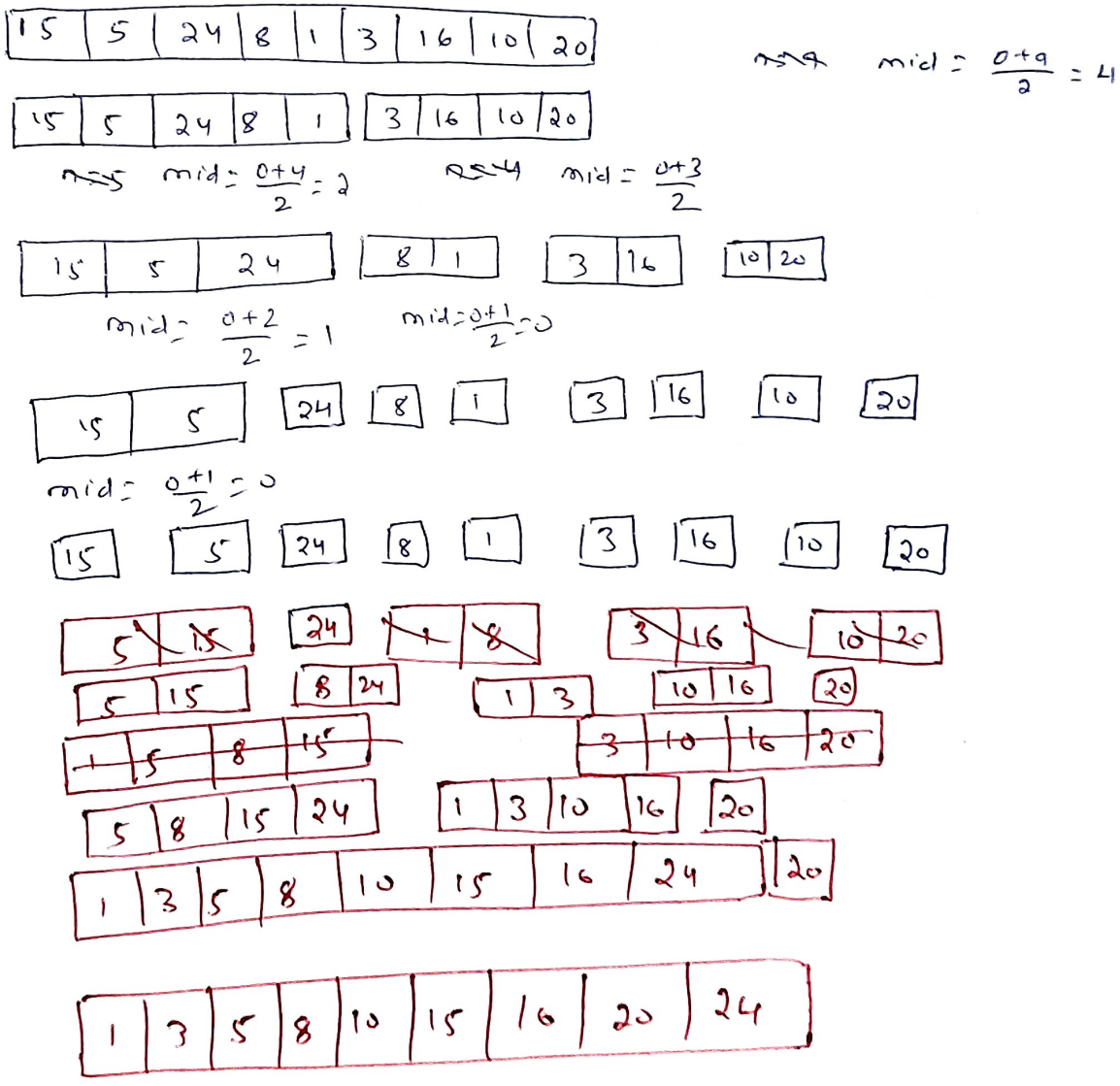
→ It divides the given list into two equal halves and again divides the sub array into two two equal halves

→ The process will continue until only one item remaining in a sub array.

→ Then take one sub array & compare it with another subarray and sort it.

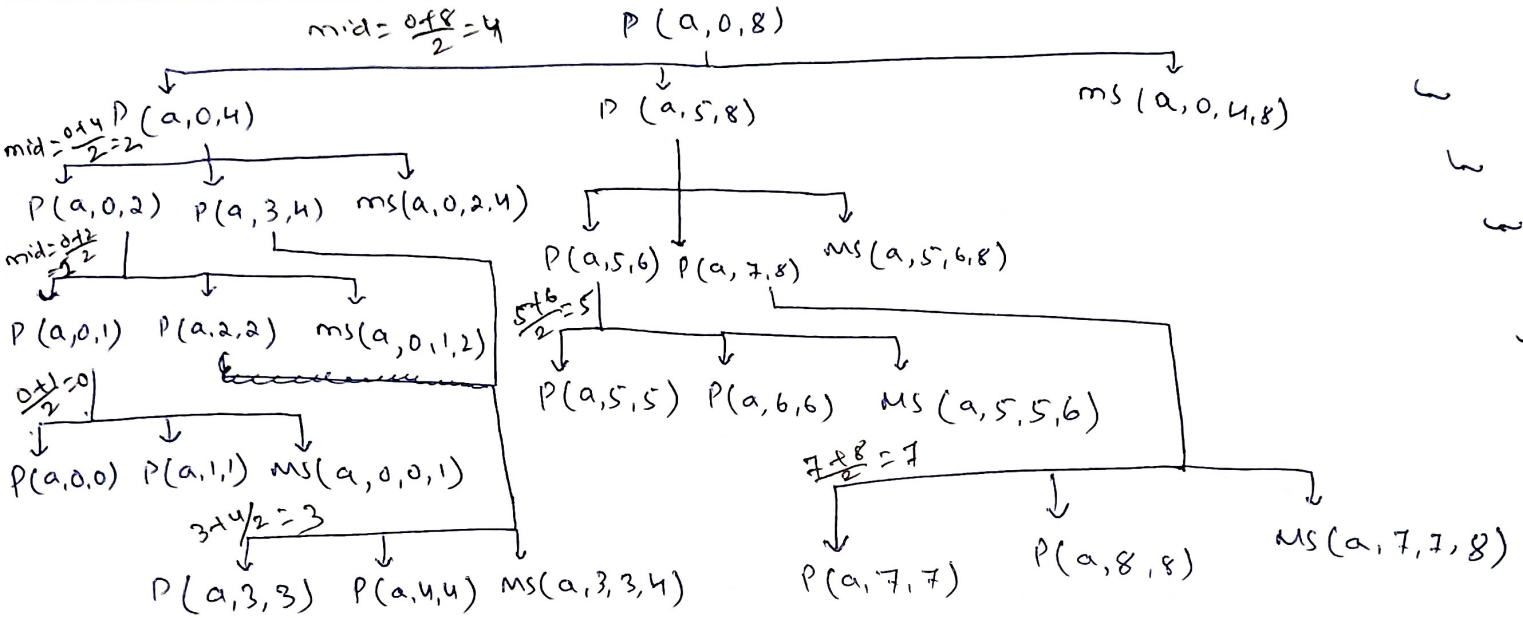
→ Continue the process until the array got sorted.

Ex:



Code:

```
Class MergeSort{
    Public static void main (String[] args) {
        int n, i;
        int a[] = {15, 5, 24, 8, 1, 3, 16, 10, 20};
    // n = sc.nextInt();
        n = a.length;
        Partition (a, 0, n-1);
        for (i=0; i<n; i++)
            S.o.p (a[i] + " ");
    }
    Static void Partition (int []a, int low, int high) {
        int mid;
        if (low < high) {
            mid = (low + high)/2;
            Partition (a, low, mid);
            Partition (a, mid+1, high);
            mergeSort (a, low, mid, high);
        } }
    Static void mergeSort(int a[], int left, int mid, int right) {
        int i, j, k, s1, s2;
        s1 = mid - left + 1;
        s2 = right - mid;
        int left_arr[] = new int [s1];
        int right_arr[] = new int [s2];
        for (i=0; i<s1; i++)
            left_arr[i] = a[left+i];
        for (j=0; j<s2; j++)
            right_arr[j] = a[mid+1+j];
        i=0; j=0; k=left;
        while (i<s1 && j<s2) {
            if (left_arr[i] <= right_arr[j]) {
                a[k] = left_arr[i];
                i++;
            }
            else
                a[k] = right_arr[j];
                j++;
            }
```




```
        k++;
    }
    while (i<=s2)
    {
        a[k] = right_arr[j];
        j++;
        k++;
    }
    while (i<=s1) {
        a[k] = left_arr[i];
        i++;
        k++;
    }
}
}
```

- The program is done by recursion.
- As we know recursion executes from bottom to top.
- So here Partion(a,0,0), Partition(a,1,1), mergesort(a,0,0,1) will execute first.
- Then P(a,0,1), P(a,2,2), ms(a,0,1,2) will execute.
- Like this the process continue until P(a,0,8) and the array will get sorted.

ARRAYS CLASS IN JAVA

-> Arrays class is a part of the Java collection framework in the java.util package.

-> It only consists of static methods and methods of object class.

-> It is a dynamic array.

-> We can perform sorting, searching, filling, comparision of two arrays and many more operation using predefined method of Arrays class.

Pre-defined methods of Arrays class

Java Arrays class have some predefined method as follow;

(1) Sort()

-> It is used to sort the array.

-> We can use this method to sort any datatype array (both primitive and non-primitive).

-> Ex:

```
int a[] = {10, 15, 8, 7, 5};
Arrays.sort(a);
for (int e:a)
    S.o.p.(e+" ");   // 5 7 8 10 15
```

We can view output as array format by using string ie.

```
S.o.p(Arrays.toString(a));   // [5, 7, 8, 10, 15]

String s[] = {"rohit", "virat", "dhoni", "rahul"};
Arrays.sort(s);
S.o.p(Arrays.toString(s));   // [dhoni, rahul, rohit, virat].
```

binarySearch()

-> Searching of Arrays follows the binary search technique.

-> For binary search, prerequisite is sorted array.

-> So we have to first sort the array and the search, but during binarysearch, compiler automatically sort the array.

Ex:

```
int [] a = {1, 3, 2, 4, 6, 5};
Arrays.sort(a);
Arrays.binarySearch(a, 4);   // If the element is present in array, it will return its index.
Arrays.binarySearch(a, 7);   // If the element is not present in array it will return the index where the element can place by (-(index+1))
Arrays.binarySearch(a, 1, 4, 7);   // It will return the index of 7 by checking between 1 to 4 and return index based on the presence of 7 between 1 to 4. If 7 is present bet' 1 to 4 then return index else return where it can be placed bet' 1 to 4.
```

fill()

→ This function is used to fill the array with a value.

Ex:

```
int a[] = new int[5];
Arrays.fill(a,6); // It is used to fill the array by 6.
S.o.pln(Arrays.toString(a)); // [6,6,6,6,6]
int b[] = new int[5];
Arrays.fill(b, 2, 4, 6); // It is used to fill the array by 6 from
S.o.pln(Arrays.toString(a)); // [0,0,6,6,0]
```
2 to 3 (excluding 4).

copyOf()

→ It is used to make a copy of an array by increasing or decreasing the size.

Ex:

```
int a[] = {1,2,3,4,5};
int b[] = Arrays.copyOf(a,6); // [1,2,3,4,5,0]
int c[] = Arrays.copyOf(a,3);
S.o.pln(Arrays.toString(b)); // [1,2,3,4,5,0]
S.o.pln(Arrays.toString(c)); // [1,2,3]
```

copyOfRange()

→ It is used to make a copy of array ~~from~~ between a given range excluding the upper range.

→ Ex:

```
int a[] = {1,2,3,4,5};
int b[] = Arrays.copyOfRange(a, 2, 5);
S.o.pln(Arrays.toString(b)); // [3,4,5]
int c[] = Arrays.copyOfRange(a, 3, 9);
S.o.pln(Arrays.toString(c)); // [4,5,0,0,0]
```

→ If no values present in the given range, then it will filled by '0'.

equals()