Path to placements :-

Step 1 :- Language

Step 2 :- DSA

Step 3 :- projects, theory subjects, Resume, referals, A + C.I.P

$ Flowcharts :-

Diagram to represent solutions of problems

↓

1) Small parts
2) logically arrangement

① Tea start

1) water in bowl
2) stove on
3) sugar, milk
4) boil
5) exit

$ Components :-

Start / Exit → Ellipse / (Vowel) shape.

Input / output → Read n → Parallelo - gram

process → rectangle

Decision → Rhombus / Diamond

yy / no

↓ Arrow.

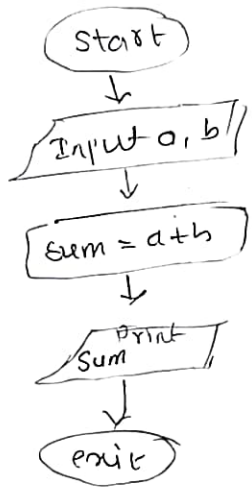
used to represent the flow.

* Sum of two Numbers :-

input :- first no, A
second No, b

output :
Sum of a & b

Flowchart



pseudo code

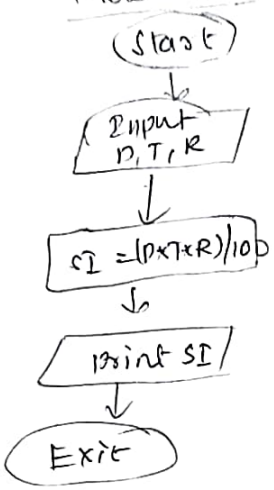
1) start
2) input no, a, b
3) sum = a+b (calcul
4) print sum
5) exit

Calculate Simple Interest

Input :-
Principal, P
Time, T
Rate, R.

Output :-
(P×T×R)/100

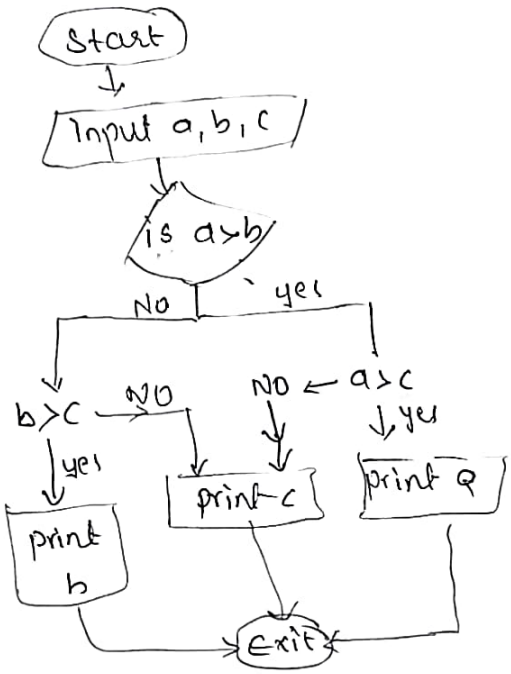
Flowchart-



Pseudocode

1) start
2) Input P T R
3) SI = PTR/100 calculate
4) print SI
5) Exit

Find Max of 3 numbers :-



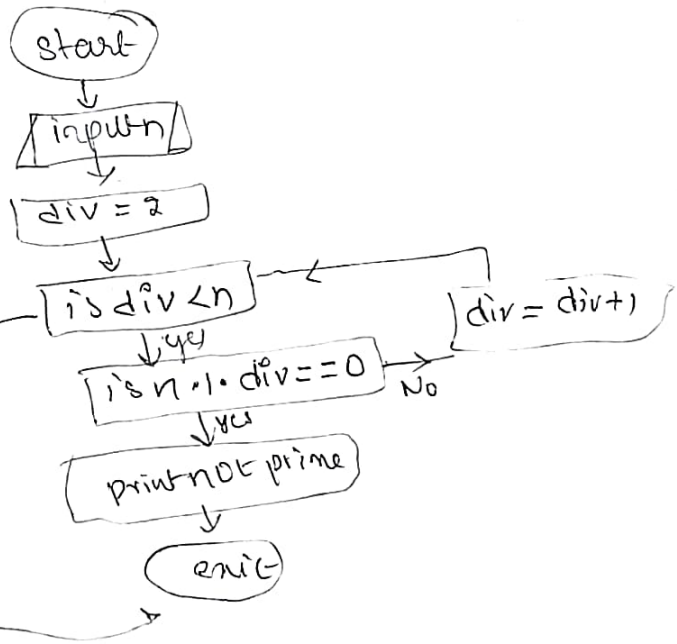
pseudocode

1) start
2) input a, b, c
3) if a>b
if (b>c)
print ("a")
else
print c
else
if b>c
print b
else print (c)
4) exit

Find prime numbers :-

pseudo code :-

1) start
2) input n
3) let div = 2
4) while div < n do
if (n%. div == 0) do
print (not prime)
exit
else
div = div+1
5). print prime
6). exit



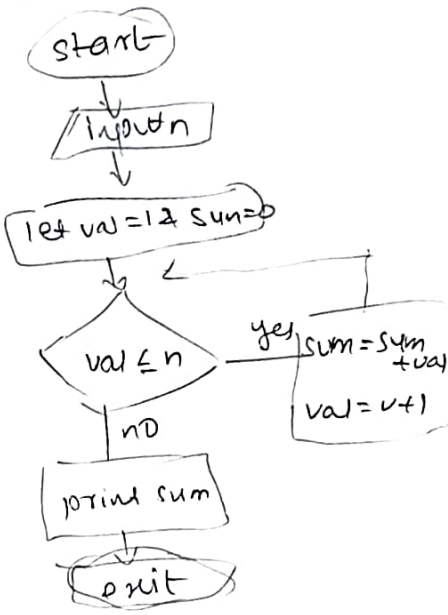
Sum of first ... natural numbers

-> natural Number (1 $\xrightarrow{to}$ N)
-> whole numbers (0 to N)

pseudo code :-

1) start
2) Input
3) Let val = 1 & sum = 0
4) while val $\leq$ n do
   sum = sum + val
   val = val + 1
5) print sum
6) Exit

Flow chart-



Flow chart Question (Assignment) :-

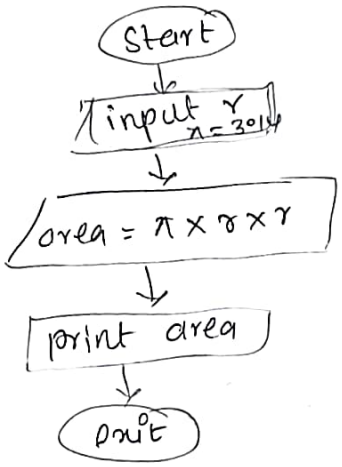
Calculate area of circle.

D/f = a.o.c = πr^2

1) pseudo code

1) start
2) input r
3) assign $\pi = 3.14$
4) area = $\pi \times r \times r$
5) print area
6) Exit.

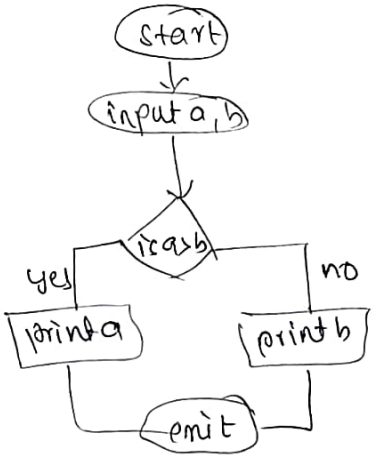
flow chart



Find the greatest from 2 numbers.

pseudo code :-

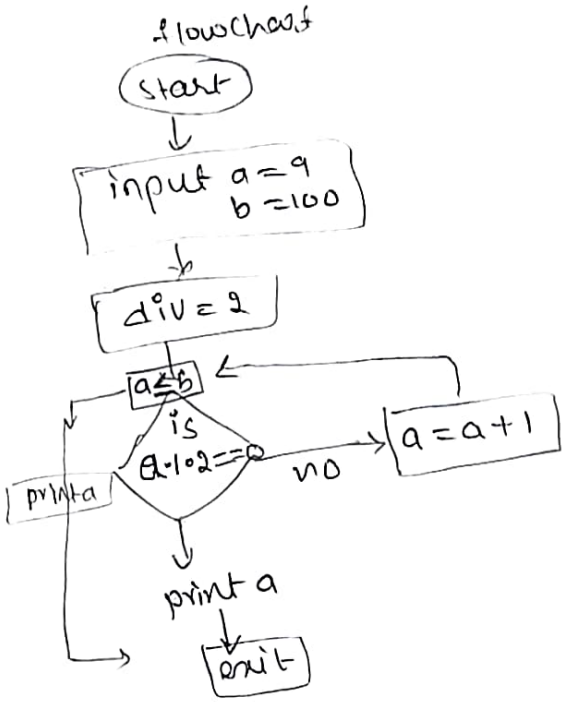
1) start
2) input a, b
3) if a > b
   print a
   else
   print b
4) exit.



uestion 3:- flowchart to print even

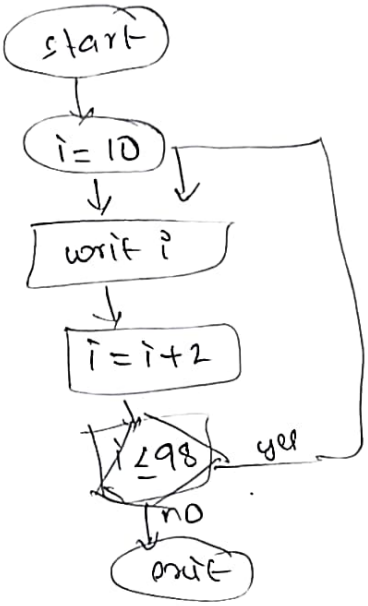
Pseudo code :-

1) start
a) input a=a b=100
div = 2.
3) while a ≤ b
if a%div == 0
print a.
a = a+1
else
3) if a%div == 0
print a.
else
a = a+1



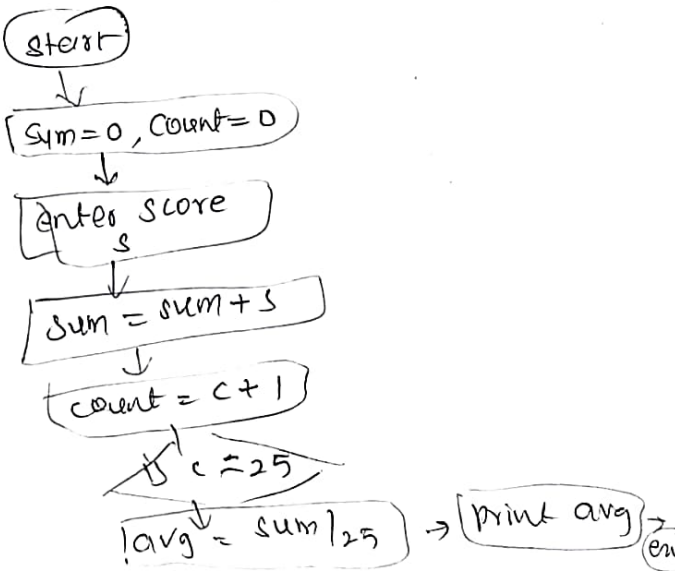
pseudo code :-

1) start
2) i = 10
3) write i
4) i = i+2
5) if i ≤ 98
write i
else
exit
6)



Q4:- find the average of 25 subject marks.

1) start
2) sum=0, c=0
3) enter score, score
4) sum = sum + score
5) count = c+1
6) if c = 25
then
avg = sum/25
else
exit
7) print avg
8) exit.



```
public static class className {
    public static void main(String args[]) {
        // code System.out.print("something");
    }
}
```

(capital) (small) (small)

Boilerplate code.

output in java :-

System.out.print("Hello");

output

function ("something");(Termination)

1) print → S.O.P.("H.W");

2) println (output in newline) → S.O.Pln("H.W

or

print("Hello \n")

(new line)

2) "\n" → S.O.P("H.W\n");

print pattern :-

```
* * * *
* * *
* *
*
```

* sten.

```
S.O.Pln("* * * *");
S.O.Pln("* * *");
S.O.Pln("* *");
S.O.Pln("*");
```

Variables in java :-

a = 10, b = 5

2 * (a+b) // 2x1xh

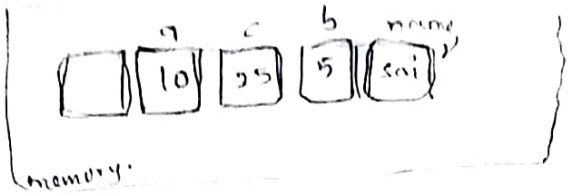
literal / universal (variable)

a(2) b(5)

// variable declaration :- int (variable name) = (value)

int a = 5;

Memory:-




```
int a = 10;
Int b = 5;
int c = 25;
string name = "sai";
```

Code:-

```
public class Basics
    {
    P S V M ( string Args [])
     {
     int a = 10;
     int b = 5;
      S.O.println(a);
      S.O.println(b);
    }
}
```

Data types in Java

↓

primitive (existed)

byte → 1 byte (size) → 8
short
char
boolean
int
long
float
double

Non-primitive (Userdefined)

String
Array
class
Object
interface.

ex:- Code:

```
P S V M (S.A[])
 {
1) int a = 10;
2) byte b = 8;
3) char ch = 'a';
4) boolean var = true;, false;
5) float a = 10.2;
6) long → long int
7) double → double (long float)
8) short u = 240;
```

size of Data type.

byte → 1 byte [-128 to 127] 256
short → 2 byte
char → 2 byte
int → 4 byte
bool → 1 byte [true, false]
float → 4 bytes [-2 Billion to +2 billion]
long → 8 bytes [.]
double → 8 byte

Sum of a & b,

```
int a = 10;
int b = 20;
int sum = a+b;
S.O.print('sum');
```

Comments in Java

// something (single line)

/* ..... */ (multiple line)

* → asterisk

Input in Java :-

- → next
- → nextLine
- → nextInt
- → nextByte
- → nextFloat
- → nextDouble
- → nextShort
- → nextLong ..

To take input we need to create a object

→ scanner [class]

→ Scanner sc = new scanner (sy.in

→ import java.util.*;

StackOverflow → solutions for errors website.

Sum of a & b (user Input),

```
import java.util.*
public class JavaBasic {
    psvm (S*[])
    {
        scanner sc = new scanner (system.in
        int a = sc.nextInt();
        int b = sc.nextInt();
        int sum = a+b;
        S.O.pln (sum);
    }
}.
```

product of a & b :

```
scanner sc = new scanner (system.in);
int a = sc.nextInt();
int b = sc.nextInt();
int product = a*b;
S.O.pln (product)
```

Area of Circle :-

Area = πr^2

```
psvm(s*[])
{
Scanner sc = new Scanner(System.in);
float rad = sc.nextFloat();
float area = 3.14 * rad * rad;
S.O.pln(area);
}
}
```

3.14 = double

3.14f = float

Type conversion :-

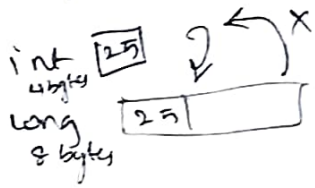
conversion happens when:

a) type compactible

b) destination type > source type

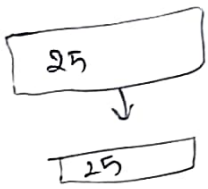
ex:- int = 25

long = 25



→

byte → short → int → float → long → double.



(Lossy conversion)

long to int data is lossed.

ex:-

```
Scanner sc = new Scanner(S.in);
int n = sc.nextFloat();
S.O.pln(n);
```

→ int cannot convert to float

can

6 → 6.0 ✓

6.0 → 6 ✗

float → int ✗

Type casting :- Explicit (small to big) char to int

Implicit (big to small) long to int

```
float a = 25.0;
int b = a;  (loss of data)
⇧
int b = (int)a;
```

```
Scanner sc = new Scanner(S.in)
float a = 25.12f;
// int b = a;  // Lossy conversion
   int b = (int) a;
   S.O.pln(b);
   o/p = 25
```

```
float n1 = 99.999f
int n2 = (int) n1;
S.O.pln(n2)
   o/p = 99.
```

→

```
char ch = 'a';
int number = ch;
S.O.pln(ch);
   o/p = 97
```

```
char ch = 'b'
int n = ch;
   S.O.pln(ch);
   o/p = 98
```

Type promotions in Expressions :-

1) Java automatically promote byte, short, or char operant to int when evaluating an expression.

2) if one operand is long, float or double to whole expression is converted to long, float & double.

1) → byte, short, char

ex:-

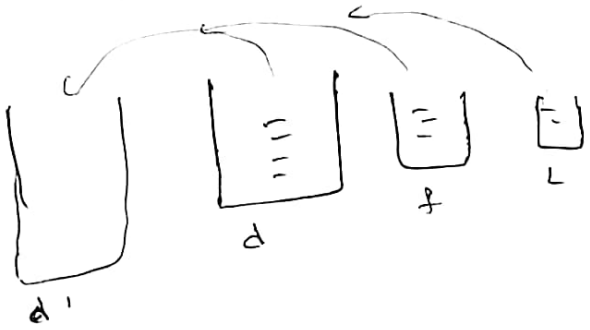
```
char a = 'a';
char b = 'b';
S.O.pln(int(a));
S.O.pln(int(b));
S.O.pln(b-a);
```

```
o/p = 97
      98
      1.
```

```
short a = 5;
byte b = 25;
char c = 'c';
// byte bt = a+b+c;
   (Lossy conversion)
byte bt = (byte)(a+b+c);
```

2) long float double.

```
int a = 10;
float b = 20.25f;
long c = 25;
double d = 30;
double ans = a+b+c+d;
System.o.pln(ans);
```



ex :-

```
byte b = 5;
byte b = b*2;   // Lossy conversion
S.o.pln(a);
```

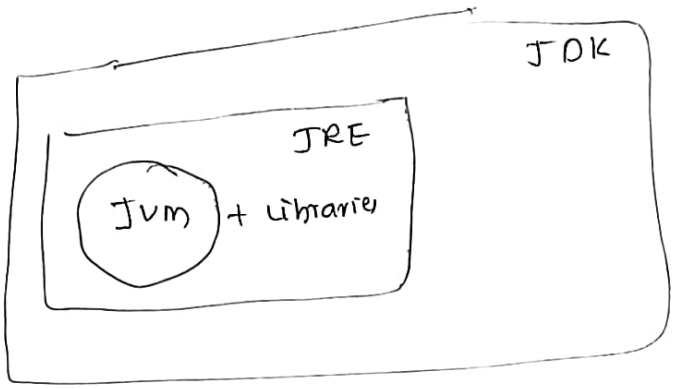
byte b = (byte)(b*2); ✓

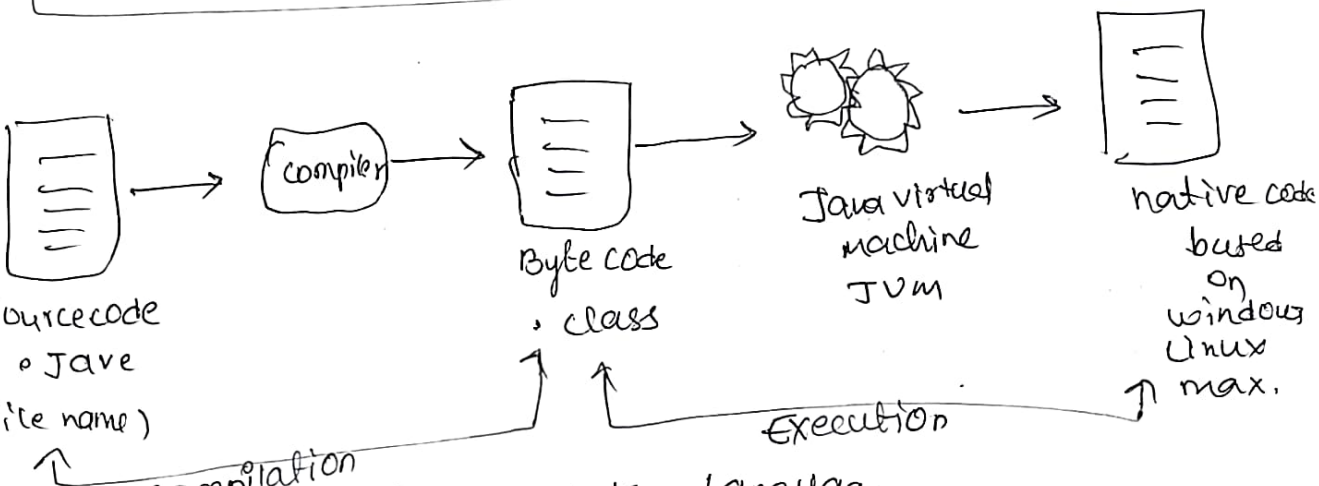
byte ← byte ← int

How code runs in System

JRE → Java runtime environment

JDK → Java development code.





Java is a portable language

(works on any system (windows linux mac.

6. operators

Symbols that tell compiler to perform some operations.

```
Sum = a + b
      ↓ └→ operator.
   operand   expression
```

Arithemetic operators (binary/unary) Ternary)
Relational operators
Logical operators
Bitwise operators
Assignment operators

1) Arithemetic operators

Binary
+
−
*
/
%

Unary:
++ (increment)
-- (decrement)

```
s.o.pln (A+B);
(A-B);
(A*1.B);
(A*B);
(A/B);
```

pre increment
++a
1) value change first
2) use value

```
int a = 10;
int b = ++a;
s.o.pln(a);
s.o.pln(b);
o/p = 11
      11
```

post increment
a++
1) value use
2) value chang

```
int a=10
int b= a++
s.o.p(a)
s.o.p(b)
op1 11
    10
```

pre decrement
--a
1) value change
2) use value

```
int a = 10
int b = --a
s.o.pn(a)
s.o.pn(b)
= 9
  9
```

post decrement
a--
2) use value
3) change value.

```
a=10
b= a--
s.o.p(a)
s.o.p(b)
= 9
= 10
```

Relational operators

==
!=
>
<
>=
<=

a = b → Assignment operator

a == b → relational operator

= (false)

10 == 10 (True)

int a = 10
int b = 10
S.o.pln ((a==b)); true
S.o.pln (a != b); false
(a > b); true/false
(a < b)
(a >=)
(b <= a)

Logical operators

&& (logical And)

|| (logical OR)

! (logical NOT)

&& (And)

Ans	S1	S2
T	T	T
F	T	F
F	F	T
F	F	F

s.o.p((3>2) (5>2))

|| (OR)

Ans	S1	S2
T	T	T
T	T	F
T	F	T
F	F	F

s.o.pln ((5>3)||(3<5))

o/p true

Logical NOT :- (!)

true → false (! true)
false → true (! false)

s.o.pln (!(3>2));

false.

Assignment operators :-

A = B

=	A = A + 1 (A += 1)
+=	A = A - 1
-=	A = A * 1
*=	A = A / 5
/=	

Conditional Statements :-

→ if else

→ if (conditn)
{
...
}
else {
...
}

```
int age = 22;
if (age ≥ 18) {
    s.o.p("Adult");
}
else {
    s.o.pln("child");
}
```

Conditional statements :-

if, else →
else if
ternary operator
switch.

Print the largest of 2.

```
int A, B;
if (A > B) {
    s.o.pln("A");
}
else {
    s.o.pln("B");
}
```

print if a number is odd or even :-

```
int a = 10;
if (a % 2 == 0)
{
s.o.p("even");
}
else {s.o.p("odd");
}
```

Else if :-

```
if (Condition 1) {
    // code
}
else if (cond 2) {
    // code
}
else {
    // code
}
```

Print the Largest of 3 Numbers :-

A = 1 B = 3 C = 6

```
if (A >= B) && (A >= C)
    print(A)
else if (B >= C)
    print(B)
else
    print('C')
```

Code :- Income Tax Calculato

1) income < 5L
   0% tax
2) income b/w 5-10L
   20% tax
3) income > 10L - 30%

Code :-

```
Scanner sc = new Scann
int income = sc.nextInt
    int tax = 0;
if (income < 500000)
    tax = 0;
}
else if (income >= 5L && I
    tax = (int)(income x
}
else
{
tax = (int)(income * 0.
}
s.o.pln("tax" + tax)
```

Ternary operator :
(3 operands)

Value = condition ? stmt : stmt 2 ;

int integer = (5>3) ? 5 : 3
T F

String type = (5%2==0) ? 'even' : 'odd'
T F

code :-

```
int n = 4;
// ternary oper
String type = ((n%2==0) ? "even" : "odd");
System.out.println(type);
```

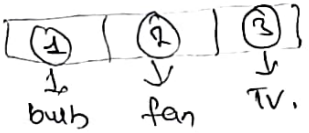
true / false

check weather student will pass/fail :-

marks ≥ 33 → pass
marks ≤ 33 → fail.

```
Scanner sc = new Scanner(System.in)
int n = sc.nextInt();
int n2 = (n≥33) ? "pass" : "fail";
S.O.println(n2);
```

Switch statement :-



syntax :-

```
switch (variable)
{
case 1 : break
case 2 : break
case 3 : break
default : break
}
```

```
int number = 2;
switch(number):
{
case 1: syso("A");
        break
case 2: syso("B");
        break.
  :
default: syso("X");
}
```

```
switch ch = 'a';
switch(ch): {
{
case "a": syso("1");
        break
case 'b': syso("4");
        break
default: syso("");
}
```

Calculator:-

```
scanner sc = new scanner(s.in)
int a = sc.nextInt();
int b = sc.nextInt();
char oper = sc.next.charAt(0);
switch(oper) {
    case 1: syso("a+b);
            break;
    case 2: syso(a*b);
            break;
      :
    default: syso("invalid oper");
}
```

[strong logic will beat everything]

Loops:-

To repeat a block of code.

1) while loop
2) for loop
3) do while loop

While:-

Syntax:-

```
while (condition)
{
// code
}
```

Print "I love 100 times".

```
int i = 0;
while (i < 100)
{
S.o.p("I love");
  i = i+1; //or
}  i++
S.o.p("printed 100 times");
```

Infinite loop:-

```
int i = 0;
while (true)
{
S.o.pln i++
}
```

- Infinite loop - system crashes

* print nos from 1 to 10;

```
int i = 1;
while (i <= 10)
{
S.o.p(i);
  i++;
}
```

* Printe numbers from 1 to n;

```
scanner sc = new scanner
int counter = 1
while (counter <= n)
{
s.o.pln(counter)
counter++
}
```

* Print sum of 1st n natural numbers.

n = 5 = 1+2+3+4+5 = 15

```
int n = sc.nextint();
int i = 1;
int sum = 0;
while (i <= n)
{
sum = sum + i;
i = i+1;
}
S.o.pln(sum);
```

Dry run:-

sum = 0
i = 1
i <= n (n=5) i++ ✓
sum = 1
i = 2
i <= n
sum = 1+2
i = 3
sum = 1+2+3
i = 4
:
upto n

* FOR LOOP

Syntax:-

```
for(initialization; condn; update)
{
// code
}
```

Hello world → 10 times

```
for(int i=0; i<10; i++)
{
s.o.p("Hello");
}
```

print square pattern (4×4)

→ 4 rows
→ 4 columns

```
int i = 1
for(int i=1; i<=4; i++)
{
s.o.pln("xxxx")
}
```

o/p

x x x x
x x x x
x x x x
x x x x

```
int i=1;
while(i<=4){
s.o.pln("xxxx");
i++
}
```

print reverse of a number

n = 10899

n = 132/10 = 2 (last digit)

→ To get last digit

num % 10

→ To remove last digit

num / 10

Code :-

```
int n = 10899;
while(n>0)
{
int Lastdigit = n%10
s.o.pln(Lastdigit);
n = n/10;
}
s.o.pln() + next line.
```

code :-

```
int n = 10988
int rev = 0;
while(n>0)
{
int Lastdigit = n%10;
rev=(rev x 10) + Last digit;
n=n/10;
}
s.o.pln(rev)
```

←

reverse

rev = (rev×10) + lastdig

n=10899 rev = 0

LD=9 rev = (0×0)+9 = 9

n=1089 → rev = (9×10)+9 = 99

108 → rev = (99×10)+8 = 9

10 → rev = (998×10)+0 = 99

1 → rev = (9980×10)+1

= 99801

do while loop

```
do {
  some work
}
while (condition);
```

```
do {
  sop("Hw");
}
while (i<=10);
```

Break Statement

- to exit the loop

```
for (int i=1; i<=5; i++)
{
  if (i==3)
  {
    break;
  }
  System.o.p(i);
}
S.O.pln("Outer loop");
```

*:- Keep entering until we get 10 multiple

```
Scanner sc = new Scanner(S.in)
  int
    do {
    int n = sc.nextInt();
    if(n%10==0) break
      S.O.pln(n);
    }
    while(True);
```

```
do {
  int n = sc.next();
  if (n%10==0)
  {
    break;
  }
  so.o.pln(n);
}
while (True);
```

Continue Statement

To skip an iteration.

```
for (int i=1; i<=5; i++)
{
  if (i==3)
  {
    continue;
  }
  S.O.p(i)
}
S.O.p("Outer loop")
```

* Display all numbers entered by user except 10 multiple

code:- scanner sc = new scanner

```
do
{
  int n = sc.nextInt()
  if (n%10==0)
  {
    continue
  }
  S.O.pln(n)
}
while (True);
```

check if a number is prime or not

prime = 2, 3, 5, 7, 11 . . .

```
Scanner sc = new Scanner(S.in)
int n = sc.nextInt();
boolean isprime = true; // Assume first
for(int i=2; i<=n-1; i++)
{
if (n%i==0) // multiples are there or not
                n is a multiple of i (i ≠ 1 or n)
  {
  isprime = false // updation
  }
}
if (isprime == true)
  {
  S.O.pln("prime");
  }
else
  {
  S.O.pln("Not prime");
  }
if (n==2) { S.O.p("prime"); }
```

$n = \sqrt{n} \times \sqrt{n}$

for Larger numbers

```
for (int i=2; i<=n-1
      i++
for(ini=0; i<= math.
      sqrt(n)
```

```
for(i=2; i<= math.Sqrt(n
math.sqrt();
```

present in util.*;

Nested loops

→ loops under loops

```
if (a>b)
  {
  if (a>c)
    {
    // code
    }
  }
```

```
for (int i=0; i<n; i++)
  {
  for (int j=0; j<n; j++
    {
    u code
    }
  }
```

Pattern :-

* → 1 (1) - 1 time
* * → 2 (2) - 2 time
* * * → 3 (3) - 3 time
* * * * → 4 (4) - 4 time

Step 1 :- How many lines.
(outer loop has to run) - n times.

(1) line (4)
Outer loop → n times,

(2) no of times
Inner loop (i times)

(3) what to print
* "

Step 2 :- In line (inner loop)
(how many time has to run)

Code :-

```
for (int line = 1 ; line <= 4 ; l++)
    {
for (int j = 1 ; star <= line ; j++)
        {
            syso ("*");
        }
        sy.println();
}
```

Dry run :-

line	Star
v = 1	1 ≤ 4 *
v = 2	2 ≤ 4 * *
v = 3	3 ≤ 4 * * *
v = 4	4 ≤ 4 * * * *
5 = 5	5 ≤ 4 x

o/p :

```
*
* *
* * *
* * * *
```

Inverted star pattern :- stars = n - i + 1

* * * * → L1 (4)
* * * → L2 (3)
* * → L3 (2)
* → L4 (1)

```
int n = 4;
for (int i = 1 ; i <= 4 ; i++)
    {
for (int j = 1 ; j <= n-i+1 ; i++)
        {
            s.o.p("*")
        }
}
s.o.println();
```

print ~~half~~ pyramid :-

```
1
1 2
1 2 3
1 2 3 4
```

① Outer loop = 4 lines

② inner loop

```
1,      i=1 ,  1        1 to 1
1 2     i=2 ,  1 2      1 to 2
1 2 3   i=3 ,  1 2 3    1 to 3
1 2 3 4 i=4 ,  1 2 3 4  1 to 4
```

③ print number (inner loop count)

code :-

```
for (int i=1 ; i<=4 ; i++)
  {
  for (int j=1 ; j<=i ; j++)
     {
     syop(j);
     }
   s.o.println();
  }
```

Dry run

			print
i=1	j=1	1<=1	1
i=2	j=1,2	1<=1, 2<=1	1 2
i=3	j=1,2,3	4<=3	1 2 3
i=4	j=1,2,3,4	5<=4	1 2 3 4

```
1
1 2
1 2 3
1 2 3 4
```

print chacter pattern :-

{ continuation }

```
i=1, char=1   A
i=2, c=2      B C
i=3, c=3      D E F
i=4, c=4      G H I J
```

① outer loop = 4 lines
n=4

② inner loop

char ch = 'A', 'B', 'C', 'D', 'E', ...

update ch++

```
int n=4;
char ch='A';
for (int i=1 ; i<=n ; i++)
  {
  for (int chars=1 ; char<=i ; chars++)
     {
     s.o.p(ch)
     ch++
     }
  Syso pln()
  }
```

Dry run

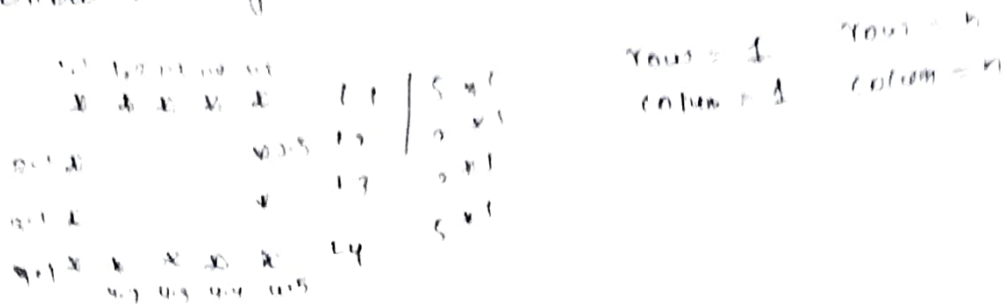
line = 1	J=1	A
	c++ = B.	
line = 2.	J=1, c++ 'C'	B C
	J=2	
line = 3	J=1 'D'	D
	J=2 'E'	E
	J=3 'F'	F

char++;

o/p =

```
A
B C
D E F
G H I J
```

Hollow rectangle



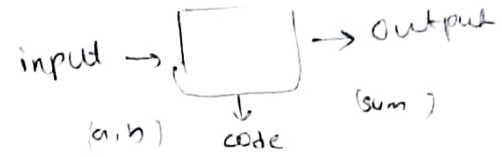
Functions / Methods

→ function :- work.
block of code use to do a task is called function

block of code / reuseable

SYNTAX :-

```
returntype name( )
    {
    // body
    return statement;
    }
```



Code :-

```
public static void printHello( )
        {
        Sysо("Hi world")
        return ;
        }
public static void main(String args[ ])
        {
        printHello( ); // function call
        }
```

syntax with parameters:-

```
return-type fname (type param1, type param2)
{
// body
return stmt;
}
```

ex:- sum of a & b

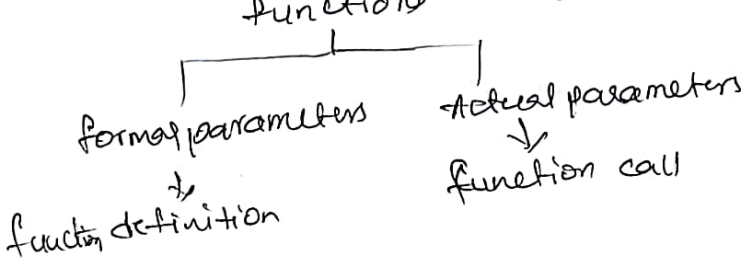
```
import java.util.*;
public static void main
sum (int n1, int n2)
{
    Sum = n1 + n2;
    return ;
}
psvm (S.A);
{
int a = sc.nextInt()
int b = sc.nextInt()
int sum z
sum (a,b) // function call
}
```

int name(int n, int n
(parameters)

name (a,b)
arguments
(actual parameters)

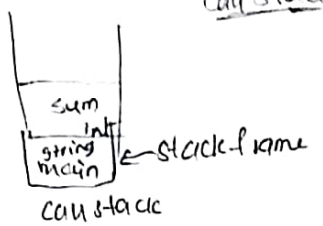
Parameters :- values passed during function creation (formal)

Arguments :- values passed during function call (actual parameter)



what happens in memory.

```
psv sum(a,b)
{
int sum = a+b;
}
psvm(SA[])
{
sum()
}
```



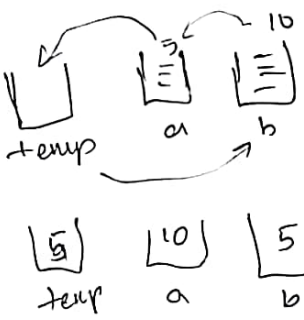
After the code is completed

The sum() & main block is eliminated from call stack.

call by value :-

Just always call by value.

during function call it gives copy value.



→ one function is written inside another function the values of variables are passed as copy values

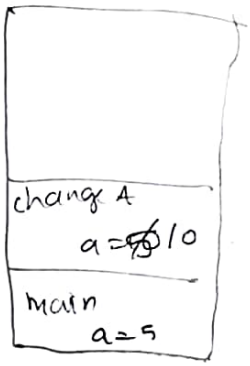
Swapping of 2 numbers;

```
int temp = a
int a = b
int b = temp;

sysO(a);
sysO(b);
```

call by reference :-

- it gives original value.

```
change A(int a)
{
a=10;
}
psvm()
{
int a=5;
change(a);
SysO(A);
}
```



Here values are original values.

Multiplication;

```
int multiply(int a, int b) {
    int pro = a*b;
    return pro;
}

intp = multiply(3,5);
SysO(p);
```

code :-

```
public static multiply (int a, int b)
    {
    int product = a*b;
    return product;

psvm(SA)
    {
    Int a = 3;
    int b = 5;
int p = multiply (a,b); // function call
    sysO (p);
        }
```

copy values are passed.

Factorial of number, n

$n! = n$ factorial

$n \times (n-1)(n-2) \cdots 1$

$4! = 4 \times 3 \times 2 \times 1$

$0! = 1!$

```
public static int fact (int n) {
        int f = 1;
    for (int i=1; i<=n; i++)
        {
        f = f * i;
        }
    return f;

    psvm(SA)
        {
    int n = SC.next();
fact = factorial(n); function call
    S.o.pln("fact");
    S.o.pln(factorial(4));
```

Binomial coefficient

$$nC_r = \frac{n!}{(n-r)!\,r!} \qquad n=5,\ r=2.$$

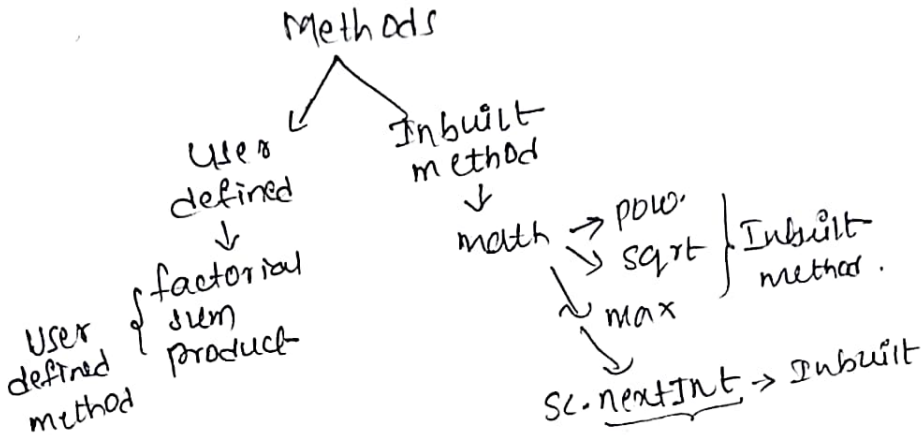
$$^5C_2 = \frac{5!}{3!\,2!}$$

```
int binocoff (int n, int r)
    {
    n_fact = factorial(n);
    r_fact = factorial(r);
n-r-fact = factorial(n-r);
```

$$BC = \frac{a}{b \times c} \qquad a=n!,\ b=r!,\ c=n-r!$$

code

```
p s int (bcoff (int n, int r)
    {
int facto_n = factorial(n);
int fact_r = factorial(r)
int fact-nmr = factorial(n-
int bincoff = fact-n / fact_r * fac n-

return bincoef;

psvm(SA)
    {
bincoeff (5,3) ||
```



Function overloading :-

multiple functions with same name but different parameters.

→ type of parameter (int, float..)
→ no of parameters (2, or 3)

ex:- calculator

```
multiply (int a, int b)
multiply (float a, float b)
multiply (double a, double b)
multiply (int a, int b, int c)
```

Function overloading using parameter :-

Same function different parameters does not depend on data type.

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

f_1 = 2 parameters

f_2 = 2 parameters

code →

```
public static int sum (int a, int b) {
    return a+b;
}
public static int sum (int a, int b, int c) {
    return a+b+c; }
    sum (3,5);    // f1
    sum (a, b, c) // f2
    4, 5, 6
```

m.function

function overloading using data-types:-

f1: to add 2 int values → int sum (int a, int b)
{
return a+b;
}

f2: to add 2 float values → float sum (float a, f.b)
{
return a+b;
}

code:

```
// f1
public static int sum (int a, int b){
        return a+b;
    }
// f2
p s float sum (float a, f b)
    {
    return a+b;
    }
//main function
syso (sum (3,5)) ; // function call
syso ( sum (3.5, 5, 0) ; // function call.
```

check if a Number is prime or not :-

① n → prime → 1, n

```
public static int boolean isprime (int n)
    {
    boolean isprime = true;
    for (int i = 2; i <= n-1; i++) {
        if (n % i == 0) {
            isprime = false;
            break;
        }
    }
    return isprime;

psvm (SA[])
{
syso (is prime (5));
}
```

(07)

```
public static boolean isprime (int n) { if (n==2); return true;
    for (int i = 2 ; i <= n-1; i++)   // only for n > 2
    {
      if (n % i == 0) {
          return false;
      }
    }
    return true;
}
psvm (S A[])
   syso (is prime (5)); // function call
```

(08) $n = \sqrt{n} \times \sqrt{n}$ (optimized function)

```
p s boolean isprime (int n) {
for (int i=2 ; i <= Math.sqrt(n) ; i++)
    {
      if (n % 2 == 0)
          return false;
    }
    return True;
⇒ if (n == 2)
    return true;

psvm(S A[])
syso (is prime (5)) // function call
```

Print all primes in range :- n=10 ⇒ 2 → 10
2, 3, 5, 7,

```
p s void PrimesInRange printinRange (int n)
    {
    for (int i=2  i<=n ; i++)
        {
      if f is prime (i) == true
        s.o print (i)
        }
      else { continue ;
        }
```

code :-

```
public static void primeInRange (int n) {
    for (int i=2; i<=n; i++) {
        if (isPrime (i)) { // true
            sysO (i);
        }
    }
    S.O.println();
}

psvm (SAC)
{
    primeInRange (20); // 2 to 20
}
```

O/P :- 2 3 5 7 11 13 17 19 upto 20.

Convert Number from Binary to Decimal :-

Binary number system → Bitwise (Bit manipulation) operator

Bit = (0, 1) computer language

Math → 0 to 9 (unique digits)

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (10 digits) dec

Decimal number system

0, 1 (2 digits) BP

Binary number system

decimal	binary
$(0)_{10}$	$(0)_2$
$(1)_{10}$	$(01)_2$
$(2)_{10}$	$(10)_2$
$(3)_{10}$	$(11)_2$
$(4)_{10}$	$(110)_2$
$(5)_{10}$	$(101)_2$
$(6)_{10}$	$(110)_2$
$(7)_{10}$	$(111)_2$
$(8)_{10}$	$(1000)_2$

Ex :- n = 1 0 1

1 0 1

1 ↓ ↓

... 2^4 2^3 2^2 2^1 2^0

$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$4 \times 1 + 2 \times 0 + 1 \times 1$

$= 4 + 0 + 1$

$= 5$

$(101)_2 = (5)_{10}$

$2^3\ 2^2\ 2^1\ 2^0$

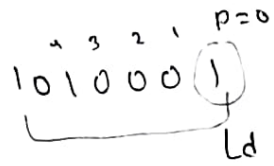
1 0 0 0

$1\times8 + 0\times4 + 0\times2 + 0\times1$

$8 + 0 + 0 + 0$

$(1000)_2 = (8)_{10}$

{ (1) Paper pen solve (process of coding)
(2) Logic
(3) code



decimal = 0

1) $dec = dec + LD \times 2^{pow} = 0 + 1(2^0)$

$LD = 1$

2) $dec = dec + LD \times 2^1 = 1 \times 1(0(2^1)$

$LD = 1$

pow = 0 1

dec = 0 1

Ex :- 1 0 1

① $dec = dec + [LD \times 2^{pow}]$

$0 + [1 \times 2^0] = 0 + 1 = 1$

② $dec = dec + [LD \times 2^{pow}]$

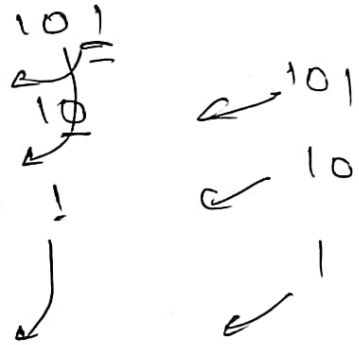
$1 + [0 * 2^1] = 1 + 0 = 1$

③ $dec = dec + [LD \times 2^{pow}] = 1 + [1 * 2^2]$

$= 1 + 4 = 5$

=

$(101)_2 \rightarrow (5)_{10}$



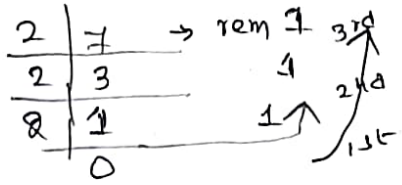
Code :-

```
public static void BTD (int binNum) {
    int myNbr = BinNbr;
    int pow = 0;
    int decNum = 0;
    while (binNum > 0) {
        int lastdigit = binnum % 10;
        decNum = decNum + (lastdigit * (int) Math.pow
                                        (2, pow)
```

```
decNum = decNum + (lastdigit * (int) Math.pow (2, pow));
    pow++;
    binNum = binNum / 10;
}
S.O.pln ("decimal of" + mynumber binNum + " = " + decNum");

psvm (SA[])
    binTodec (111); //function call ;
```

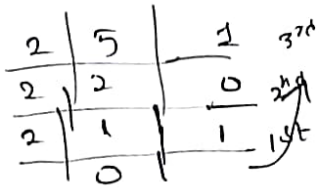
Convert from Decimal to Binary :-

n = 7



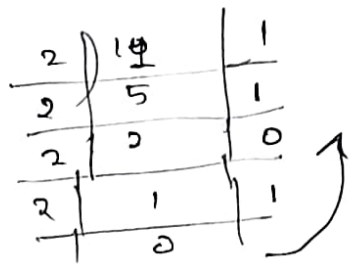
$(111)_2 = (7)_{10}$

n = 5



1 0 1
1st 2nd 3rd

n = 19



$= (10\ 011)_2 = (19)_{10}$

(while n > 0)

① devide by 2.
   ↳ rem

② bin = bin + (rem * 10^{power})

power ++

Code :-

```
public static void decTOBin (int n) {
    int pow = 0
    int binNum = 0;
    while (n > 0) {
        int rem = n % 2;
        binNum = binNum + rem * (int) Math.pow (10, pow));
        pow ++
        n = n / 2;
    }
    S.O.pln (decTOBin (19));
    // function call
```

```
public static void decToBin(int n){
    int myNum = n;
    int pow = 0;
    int binNum = 0;

while(n>0){
    int rem = n%2;
    binNum = binNum +(rem * (int)math.pow(10, pows));
        pow++;
        n=n/2;
    }
S.O.pln("binary of " + myNum + " = " + binNum);

psvm(SAC[])
    ↓
    binNum(10);
decToBin(10); // function call.
    }
```

* To get Last digit (remainder) = n%10
* To get reminder = n%2

---

scope :-

Method scope :-

↳ fun → var =

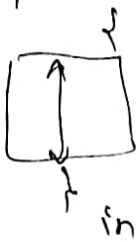
var
var } used

Code :-

```
psvm(SAC[])
    → { S.O.pln("S"); // cannot be written.
      int s = 45;
      S.O.pln(s);
    }

ps void printS(). { // scope is for only printS functi
    int s=45; }

psvm(SA[7])
    {
    S.O.p(s); // s cannot be used in main fun
        X
```

scope :

Block scope :



block of code

scope used in this block only.

Class scope

↓

Access modifiers (OOPs)

↳ public

↳ private

↳ protected

ex:-

```
psvm(S A[])
{
    int s = 45;
    s.o.println(s);
}
s.o.println(s);
```

{ block scope }

(1) scope of s is not there.

s is in { } scope

it is defined in that block only

it cannot be printed out of block { }

ex:- `for(int i=1; i<=5; i++)`

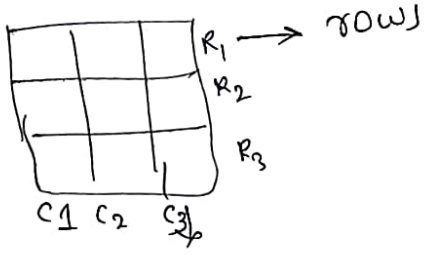
Block `{ }`

`s.o.println(i);` // i is out of block scope so it gives error.

Advance patterns

⇒ print hollow rectangle.

Matrix



size = rows x column

N x N

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4

boundary
row - 1, 4
cols 1, 5

logic:-

(1) Total line (rows) = 4
outer loop (1 to 4) if n = total rows

(2) row=1 || col = 1 || row=4 || col=5)
inner loop.

```
public static void main (S.A [])
{

}

public static void hollowrect (int rows, int cols)
{
for (int i=1 ; i<= rows ; i++)
{ // inner loop.
  for (int j=1 ; j<= cols; j++)
  { // cell - (i,j)
    if ( i== 1 || i== rows) || j== 1 || j==cols
    { // boundary
      syso(" * ")
    }
    else {
      syso(" ");
    }
  }
  syso ();
}
```

```
p s void pyramid (int rows, int cols) {
for(int i=1 ; i<=rows ; i++)
    {//inner loop
    for(int j=1 ; j<=cols ; j++)
        {//col (i,j)
        if ( i==1 || i==rows || j==1 || j==cols)
            {//boundary
            sysout ("*");
            }
        else
            {
            sysout print (" ");
            }
        }
    }
    sysoutln ( ) ;
}//first for loop braces)
}
```

rows = 4 cols = 5

Dry run :-

i = 4

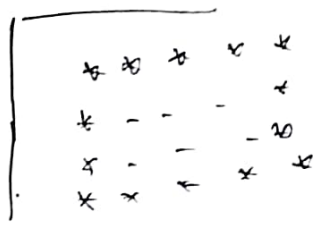
J = 5

i = 1
j = 1 2 3 4 5 6

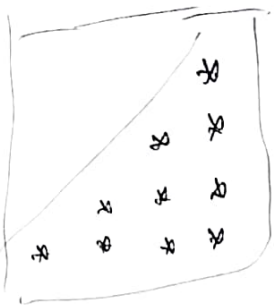
i = 2
j = 1 2 3 4 5 6

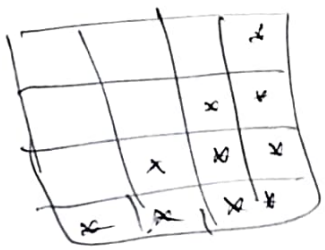
i = 3
j = 1 2 3 4 5 6

i = 4
j = 1, 2, 3, 4, 5 6
termination.



Inverted & rotated Half pyramid





① outer loop
lines = 4
for(1 → 4)

② inner loop.
spaces + stars.

r1 spaces = 3 stars = 1
r2 space = 2 stars = 2
r3 space = 1 stars = 3
r4 spaces = 0 stars = 4

rownumber = starsnumber

Row number (i) ⇒ star

Total rows = n = n − i (row)

r_1 = 4 − 1 = 4 − 1 = 3 spaces

r_2 = 4 − 2 = 4 − 2 = 2 spaces

r_3 = 4 − 3 = 4 − 3 = 1 space

r_4 = 4 − 4 = 4 − 4 = 0 spaces.

```
for (int i=1 ; i<= n ; i++ // outer loop
{
  //spaces.
 for (int j=1 ; j<= n-i ; j++
  {
  syso spaces
  }
 for (int j=1 ; j<= i ; j++)
  {
  syso("*")
  }
 } syso() // next line
```

Code :-

```
public static void pattern (int row, int cols)
{
for (int i=1 ; i<= row ; i++)
  { //spaces)
  for
```

int n

code :-

```
public static void ptn (int n)
{
for (int i=1 ; i<= n. ; i++)
  {
  //space
for (int j=1; j<= n-i ; j++)
    { syso(" ") }
for (int j=1 ; j<= i ; j++)
    { syso("*"); } syso() // next line
```

→ ptn(4) // function call

Inverted half pyramid with Numbers

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

1) Outer loop = 5 times

2) inner loop

i=1	j to 5	= n-i+1 = 5-1+1 = 5
i=2	1 to 4	n-i+1 = 5-2+1 = 4
i=3	1 to 3	n-i+1 = 5-3+1 = 3
i=4	1 to 2	n-i+1 = 5-4+1 = 2
i=5	1 to 1	n-i+1 = 5-5+1 = 1

Code:-

```
public static void invertedhalfpth (int n)
    {
    for (int i=1 ; i<=n ; i++)
        {
        for (int j=1 ; j<= n-i+1 ; j++)
            {
            sys0.pen( j );
            }
        sys0pen( );
        }
    }
public static void main (s t( ))
    {
    inverted paten (5); || function call.
    }
}
```

Floyd's triangle :-

```
1
2  3
4  5  6
7  8  9  10
11 12 13 14 15
```

1) i=1

2) i

1) outer loop : 5

2) counter increment.

```
int n = 5
int counter = 1;
pc vmptn (int n) {
for(int i=1; i≤n; i++)
{
 for (int j=1; j≤i; j++)
    {
     syso (" counter);
      counter++
    }
  syso pen ();
  }
psvm (SA{})
   {
   ptn (5); //function call.
   }
```

0-1 triangle

1				
0	1			
1	0	1		
0	1	0	1	
1	0	1	0	1

```
1
0 1
1 0 1
0 1 0 1
```

for i=1 → 1
i=2 → 0,1
i=3 → 1 0 1
i=4 → 0 1 0 1
i=5 → 1 0 1 0 1

(1,1) → 1

(2,1) 0 (2,2) 1

(3,1) 1 (3,2) 0 (3,3) 1

1 → row+co
(i+j)=
=even

① outer loop n = 5

② inner loop

for(int j = 1; j≤ i; j++)

③ if (row + col) == even
print (1)
else
print "0";

```
psv ptn (int n){
For int (i=1). i ≤ n; i++)
   {
   for (int j=1; j≤ i; j++) {
      if (i+j).%2 == 0 { //even
          syso ("1");
       else {
          syso ("0");
          }
       }
       sysopen (" ");
    }
  psvm SA{
       ptn (5); //function call.
```

Butterfly pattern :-

n=4

① First half
outer loop
for(int i=1 ; i≤n ; i++)

LS/RS ② Line

st = 1 sp=6 stars + space + stars
st = 2 sp=4
star = 3 sp=9
st = 4 sp=0

(n-i ± 4-1) (9391942937) : combine next value

sp = 6 2x3 n-i= 4-1 = 2x2=4
sp = 4 2x2
sp = 2 2x1
sp = 0 2x0

2x (n-i) = 6
2x (4-1) = 6
2x (4-2) = 4
2x (4-3) = 2
2x (4-4) = 0

1st half { Line 1 = i stars + (n-1) space + i stars
Line 2 = i s + (n-1) s + i stars

2nd half → for(int i=n ; i≥n ; i--) // reverse the for loop
for(int i=1 ; i≤n ; i++) // first half.

→ 1st half → for(int i=0 ; i≤n ; i++)
{
i *'s + (n-i) spaces + i *'s
}

2nd half for(int i=n ; i≥n ; i--)
{
i *'s + (5-i space + i *'s
}

```
Public static butterfly (int n)
{
for (int i=1 ; i≤n ; i++ ) { //outerloop
  {
    // print i-stars
  {for (int j=1 ; j≤i ; j++)
      { syso ("*");
      }
    // print spaces.
    for (int j=0 ; j<(2xn-i); j++)
      {
        syso (" ");
      }
    // print stars.
    for (int j=0 ; j≤i ; j++)
      { syso ("*");
      }
    syso ();
  }
  for (int i=n ; i≥1 ; i--)
    { for (int j=1 ; j≤i ; j++)
        { syso ("*"); }
      for (int j=1 ; j≤2x(n-i); j++)
        { syso (" "); }
      for (int j=1 ; j≤i ; j++)
        { syso ("*"); }
  syso();
PSVMSA() { butterfly (n); // function call
```

first half

second half

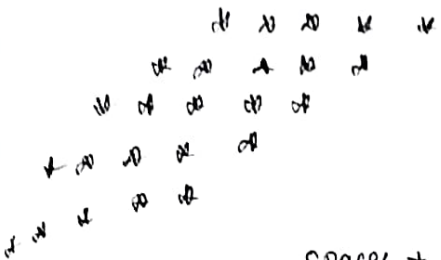
star + space + star

* — — — — *

10

3 for loops

Solid Rhombus :- (patterns and Numbers Logic)

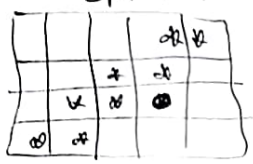


n = 5

outer loop.

1) n = 5
for (int i = 1 ; i ≤ n ; i++

(2) spaces + stars



Spaces + stars.

	(n-i)	stars = n	
i = 1	space = 4	stars = 5	5 - 1 = 4 space
i = 2	spaces = 3	stars = 5	5 - 2 = 3
i = 3	space = 2	stars = 5	5 - 3 = 2
i = 4	sp = 1	stars = 5	5 - 4 = 1
i = 5	sp = 0	stars = 5.	5 - 5 = 0

stars = n (j = 1 to j = n)

space
n - i
space.

Code :-

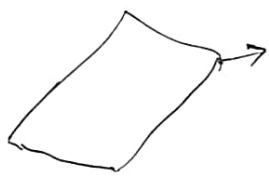
```
public static sRhombus (int n)
{
for (int i = 1 ; i ≤ n ; i++ { outer loop
  { //space
  for (int j = 1 ; j ≤ n-i ; j++
    {
      syso (" ");
    }
for (int j = 1 ; j ≤ n ; j++)
    {
    syso ("*");
    }
syso pln() //next line

P S V m S A C)
    {
    sRhombus (5); //function call
    }
```

1) outer loop
2) inner loop
spaces + stars
j = 1 j ≤ n-i j++
syso (" ")

j = 1 j ≤ n j++
syso ("*")

Hollow rhombus :-

boundary → star printed



boundary.

(1) outer
n = 5

(2) Inner loop
spaces + boundary
rectangle
+ spaces
+ boundary

```
psvm (rpattn (int n) {
for(int i=1; i<=n ; i++) // outer loop.
    { inner loop.
for (int j=0; j<= n-i; j++)
    { // space.
    sysO (" ");
    }
for (int j=1; j<= n; j++)
    {
    if (i==1 || i==n || j==1 || j==n)
      {
      sysO ("*");
      }
    else {
      sysO (" "); //spaces.
      }
    sysO pen(); next line.

psvm sA(){
    rpatern(5);
    }
```

Diamond pattern :-

```
    *
   * * *
  * * * * *
 * * * * * * *
* * * * * * * * *
 * * * * * * *
  * * * * *
   * * *
    *
```

① outer loop - n=5

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

n-i	2i-1
i=1 sp = 3 (4-1)	st = 1
i=2 sp = 2 (4-2)	st = 3
i=3 sp = 1 (4-3)	st = 5
i=4 sp = 0 (4-4)	st = 7

Odd

2x+1 | 2x-1

i=1 | 2(1)+1 = 3 | 2(1)-1 = 1

2(0)+1 = 1 | 2(1)-1 = 1

2(1)-1 = 1
2(2)-1 = 3
2(3)-1 = 5
2(4)-1 = 7

```
public static void diamond(int n)
{ //first half
{ for(int i=1 ; i<=n ; i++)
  { //spaces
  for(int j=1 ; j<=n-i ; j++)
      {
      sysO(" ");
      }
  for(int j=1 ; j<=(2i-1) ; j++)
      { //stars
       sysO("*");
      }
   sysOpen();
  for(int i=n ; i>=1 ; i--)
      {
      for(int j=1 ; j<=n-i ; j++)
       { spaces
       sysO(" ");
       }
      for(int j=1 ; j<=2i-1 ; j++)
          { //stars
          sysO("*");
          }
       sysOpen();
  }
psvm(SA[])
  {
  Rhombus(5);
  }
```

first half

second half

Pattern (QNA) Vedio Bonus live | | space ... wanted

Number pyramid :

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

1 loop + 1 loop
spaces + numbers

(1) outer loop
n = 5
1 to n
for (i=1; i ≤ n; i++)

(2) inner loop.
[n - i]

i=1 →	1	1 time	sp = 4
i=2	2	2 times	sp = 3
i=3	3	3 time	sp = 2
i=4	4	4 times	sp = 1
i=5	5	5 time	sp = 0

stars = for (int j=1 ; j ≤ i ; j++)
print (i + "_")

spaces = for (int j=1 ; j ≤ n-i ; j++)
{ syso ("space");

```
public static void ptn(int n)
{ //output, space outer loop
for (int i=1; i ≤ n; i++)
{ outer Inner
for (int j=1; j ≤ n-i; j++)
{      spaces
syso ("  ");
}
       stars
for (int j=1 ; j ≤ i; j++)
{ syso (i + "_");
}
}
syso pen(1'),
}
psvm SA()
{
ptn(5);
}
```

(Must a should)
{Next Line After every inner loop}

Palindrome pattern :- with numbers.

noon Malayalam
racecar 121
Madam 1221
       141

spaces	desending order	Ascending order.
1	1	2
2 1	2 1	2 3
3 2 1	2 1	2 3 4
4 3 2 1	2 1	2 3 4 5
5 4 3 2 1	2 1	2 3 4 5

n-i

i=1 sp=4
i=2 sp=3
i=3 sp=2
i=4 sp=1
i=5 sp=0

i + sp = n
1 + 4 = 5
2 + 3 = 5
i + sp = n
sp = n - i

① outer loop
1 to 5
for(int i=1; i<=n; i++)

② inner loop (what to print)

spaces + numbers

for(int j=1; j<=n-i; j++)
sysO(" ");

numbers

	desending loop	Ascending loop.
i=1	1 to 1	
i=2	2 to 1	2 to 2
i=3	3 to 1	2 to 3
i=4	4 to 1	2 to 4
i=5	5 to 1	2 to 5
	i to 1	2 to i

descending loop: for(j=i; j>=1; j--)

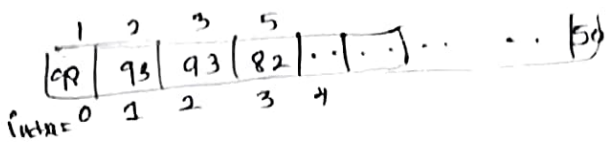
for(int j=2; j<=i; j++)

```
public static palindrome(int n)
{
// outer loop
for(int i=1; i<=n; i++)
{
// inner loop. spaces
for(int j=1; j<=n-i; j++)
{
sysO(" ");
}
// desending order
for(int j=i; j>=1; j--)
{
sysO(j);
}
// Ascending order
for(int j=2; j<=i; j++)
{
sysO(j);
}
sysO pln();
}
PSVM SA()
{
palindrome(5); // function call
}
```

Data structure

- Array :- Arrays are the data structures in which, are used to store same type of data

```
int phy:- 98
int phy = 99
int math = 93
```

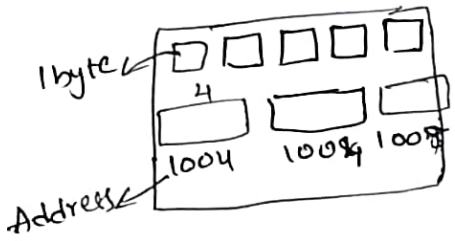


C++ / java → 0-based indexing

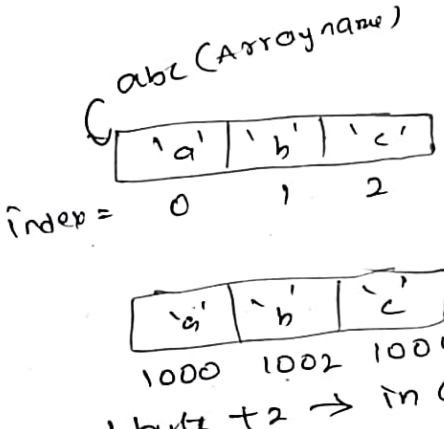
Array :- List of Elements of the same type placed in a contigious memory location

```
int array :-     1, 2, 3 . . . .
string array :-  'mango' "apple' . .
float array :    1.1, 1.2, . . . .
```

Memory :-



ex:-



1 byte +2 → in char

Operations in Arrays :-

- create
- input
- output
- update

(1) creating an Array

Syntax :-

data type arrayname [] = new datatype [size];

new keyword used to allocate memory to (location) to array.

in

```
int Marks[ ] = new int [50];
int numbers[ ] = {1, 2, 3};
int Sum[ ] = {4, 5, 6};
string fruits[ ] = {"orange", "apple", "mango"};
```

Code :-

```
public class ArraysCC {
    public static void main (S A[])
    {
    int marks[ ] = new int [50];
    }
    Pnt num[ ] = {1, 2, 3};
    String fruits[ ] = {"Oran", 'A', 'm'};

    Syso (marks());
```

→ if array is not stored any number

newint[ ] → 0 0 0 0 [by default]

'_' '_' '_' [space for strings]

if we didn't declare size of Array.

Input/output :-

```
public class Arrayc {
    psvms A {
    int marks[ ] = new int [100];
    Scanner sc = new scanner (System.in)
    marks[0] = sc.nextInt(); //phy
    marks[1] = sc.nextInt(); // chem
    Marks[2] = sc.nextInt(); // maths.
    System.out.print ("phy" + marks[0]);
                     ("chem" + marks[1]);
                     ("math" + marks[2]);
```

update → marks[2] = 100;

syso (marks[2]); // updated printed

```
marks[2] = marks[2] + 1;  // updation.

int sum = (marks[0] + marks[1] + marks[2]);

sysO (sum);
```

Array Length :- .length ⇒ Array Name . Length

```
syso (marks.length);
```

Passing arrays as arguments :-

pass by value (main function) — by reference

```
public static void update (int marks[]) {
    for (int i=0; i<marks.length; i++)
        marks[i] = marks[i] + 1;
}
```

ex:- `int marks[] = {97, 98, 99}`

```
update (marks);
for (int i=0; i<marks.length; i++)
{
    syso (marks[i] + " ");
}
syso pen();
```

Arrays arguments are passed by reference (Always)

→ change in function will change the value in main function.

```
Public static void (int marks[])   // array changed
    // update
    for (int i=0; i<arr.length; i++)
        marks[i] = marks[i] + 1
```

```
psvm sA {
    int Marks [] = {97, 98, 99}
    Array changed (marks); // function call
                → updation
```

printing

```
for (int i=0; i<arr
    syso (marks[i]
```

Space & Time complexity :-

space complexity :- is the (number) amount of space required for an Algorithm to run until complete

→ Amount of necessary space required for Algo.

time complexity :-

The amount of time required for execution of program / Algorithm.

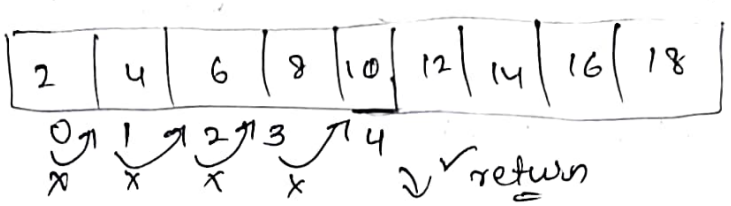
cases - **Best case**
→ fastest execution

Average case
Average execution.

Worst case
slowest execution

Linear search :-

→ find the index of element in a given array.



Complexity :- loop
for (i=0; i to n) { length

Worst case = O(n)

Time complexity = O(n)

```
Public static int linearSearch(int numberArray[], int key)
{
  for(int i=0; i<numberArray.length; i++)
  {
    if (array[i] == key)
    {
      return i;
    }
  }
  return -1; // int type returns an integer.
}
Public static void main (String[])
{
int numberArray[] = {1,2,4,6,8,10,15,106};
  int key = 10;
int index = linearSearch (NumberArray, key) // function call
  if (index == -1)                          // print output.
  {
    sysO ("Not found");
  }
  else
  {
    sysO ("Number is at" + index);
  }
}
```

LARGEST Numbers in array

find the largest number in a given array

1	6	2	4	0
0				

largest :- $-\infty$ 1 → 6 → 6 → 6 → 6 ✓ largest

$-\infty$ = integer.MIN_value

$+\infty$ = integer.MAX_VALUE

import util.*; // inbuilt library

```
public static int getLargest(int numbers[])
{
    int largest = Integer.MIN_VALUE; // -∞
    for(int i=0; i< numbers.length; i++)
    {
        if(largest < numbers[i]){      or  N > L
            largest = numbers[i];
        }
    }
    return largest;
}

psvm st []
{
    int numbers[] = {1,2,3,6,5};
    sysO("largest" + getLargest(numbers)); // function call
}
```

```
int smallest = Integer.MAX_VALUE
if(smallest > numbers[i])
{
    smallest = numbers[i]
}
```

Time complexity = O(n)

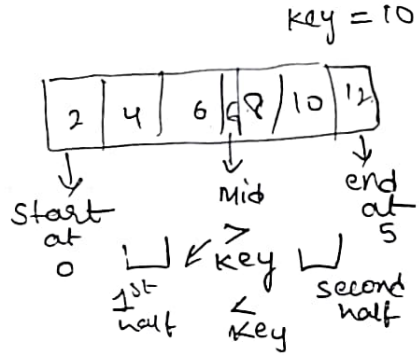
Binary Search :-

⇒ Interviews / coding rounds (Asked)

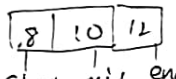
prerequisite → Sorted array
Must and should

Ascending → 1 2 3 4 5 6 7 {sorted}

descending → 7 6 5 4 3 2 1



second half =



mid = 10 return it.

1 → n

2 → n/2

⋮

upto single element of mid

pseudo code.

Time complexity of binary search

start = 0, end = n-1

while (start <= end)

find mid

mid = (start + end)/2

compare mid & key

mid = = K ⇒ found

mid > K → Left (1st half)

mid < K → Right (2nd half)

If K not found return -1;

Iteration		
1	→ $n = n/2^0$	$\frac{n}{2^k} = 1$
2	→ $n/2 = n/2^1$	$\Rightarrow n = 2^k$
3	→ $n/4 = n/2^2$	$k = \log_2 n$
4	— $n/8 = n/2^3$	$TC \propto \log_2 n$
.		
k	— $n/k = n/2^k$	
k	— $n/2^k$	

Time complexity

$TC = O(\log n)$

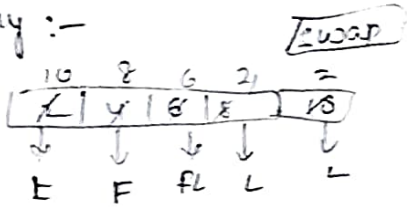
Linear search & binary search

$n < \log_2 n$

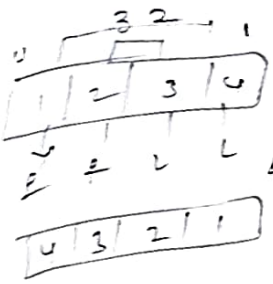
Code:-

```
public static int binarySearch(int NumArr[], int key)
{
int start = 0; end = numArr.length - 1;

while (start ≤ end)
  {
  int mid = (start + end)/2

  // comparision.
if (numArr[mid] == key)  // key = found(mid)
      {
      return mid;
      }
elseif (numArr[mid] < key) { right // 2nd half
      {
      start = mid + 1;
      }
else { // mid > key   → first half (left)
      end = mid - 1;
      }
  }
return -1; // if key is not found

PSVMS(A[]) {
    int numArr[] = { 2, 4, 6, 8, 10, 4 }
     int key = 4;
sysO (" index " + binearysearch (numArr, key));
                    o/p - index = 4
```

Reverse an array :-



Time complexity = O(n)
Space complexity = O(n)



Reversed array.

(Swap → to exchange 2 vari

Code :-

```
public static void reverse (int numArr[])
{
int first = 0; last = numArr.length - 1;
start.                end.
while (start < end) {
        //swap.
int temp = numArr [end];
numArr[end] = numArr[start];
num[start] = temp;

first++; start++;
last--; (or) end++;
```

```
psvm SA()
{
int numArr[] = {2 4 6 8 10};
reverse (numArr); // f.call
  for(int i=0; i < numbers.length; i++)
  {
  sysO( reverse numArr[i] + " " );
  }
   sysO();
}
}
```

Pairs in an Array.

2	4	6	8	10

(2,4) or (4,2) both are pairs

(2,4) (2,6) (2,8) (2,10) - 4
(4,6) (4,8) (4,10) - 3
(6,8) (6,10) - 2
(8,10) - 1

2 → 2,4,2,6,2,8,2,10 →
4 - 4,6,4,8,4,10

for (int i=0 to n)
current
for (int j = i+1 to n)
→ pairs

2/4/6
[4,6,10
[6,8,10
[8,10]

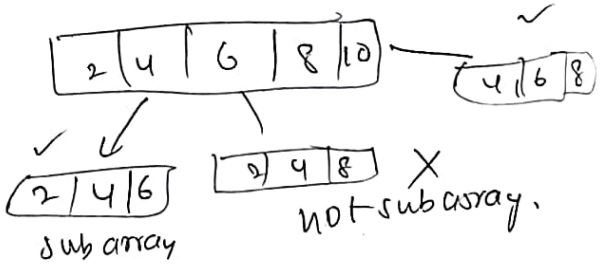
code :-

```
public static void printpairs(int Arr[])
{  int totalpairs = 0;
   for(int i=0 ; i < Arr.length ; i++)
   {  int current = numArr[i];   // 2,4,6,8,10
      for (int j= i+1; j < arr.length ; j++)
      {
         syso (" curr " + " " + numberArr[j]);
         tp++;
      }
      syso(pen();
   }
   syso(tp);

psvm s*[]
{
int numArr = {2,4,6,8,10}
printpairs(numArr); // for case
}
```

Totalpairs = $\frac{n(n-1)}{2}$ = Time complexity = $O(n^2)$ ↓ nested loop

Subarrays :-

→ A continous parts of array



2→

subarray.

2 → 2 2,4 2,4,6 2,4,6,8,10

4 → 4 4,6 4,6,8

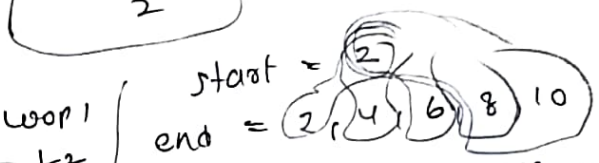
6 → 6,8 6,8,10

8 → 8,10

10 → 10

sum of n numbers.

$\frac{n(n+1)}{2}$ formula.



2,4 2,4,6 2,4,6,8 ...

start for (int i = 0 to n) — loop1

end for (int j = i+1 to n) — L2

for (start to end) — L3

}subarray.

```
public static void printSubArray(int num[])
    {  i=0;
    for(int i=0; i< num.length ; i++)
    int start = num[i]; or i
    for(int j=i; j< num.length; j++)
        {  int end = num[j] or J
        for(int k = start; k <= end; k++)
            {
            syso(number[k]+ " ");
                ts++;
            }
        syso();
        }
    syso();
}

psvm sA
{
int num[] = {2,4,6,8,10};
printsubarr.(number) // f call

}
    syso pen(ts); //total subarrays.
```

Max - Subarray sum. (Brute force) Stack.

Time complexity :-

3 for loops

$= O(n^3)$

1	-2	6	-1	3

→ 1 (1,-2) (1,-2,6) (1,-2,6,-1) (1,-2,6,-1,3)

Sum 1 -1 5 4 7

→ -2 → sub array
- 6
- -1
- 3

Sum = max ⇒ print max

ex

1	-3	2

current = 1

max-sum = $-\infty$

$-\infty < 1$

max-sum = 1

Code

```
Public static void maxSumSubarray(int arr[]) {
    int currsum=0;
    int Maxsum = Integer.MIN_VALUE;
    for(int i=0; i<Arr.length; i++) {
        int start = i;
        for(int j=0; j<arr.length; j++)
        {
        int end = j;
        currsum=0;
        for(int k=start; k<=end; k++) { //print
            //subarray sum
        currsum += arr[k];
        }
        sysOPLn(currsum);
        if(maxsum<currsum) {
            maxsum = currsum); sysO("max sub" + maxsum)
        }
PSVMS{ int arr = {1,2,3,-1,2}
        maxsubarry(arr);
```

prefix sum approach of Max subarray sum.

prefix array

1	-1	5	4	7
0	1	2	3	4

TPM Complexity = $O(n^2)$

prefix[end] - prefix[start-1] = 4-1 = 3

code :-

```
public static void maxSubarraySum (int num[])
{
    int currsum=0;
    int maxSum = Integer.MIN_VALUE;
    for(int i=0; i<numbers.length; i++){
    int
int prefix [] = new int [numbers.length];
prefix [0] = numbers [0];

    for(int i=1; i< prefix.length; i++) {
        prefix[i] = prefix [i-1] + numbers [i];
    }
for(int i=0; i< number.length; i++) {
    int start = i;
for(int j = i; j < numbers.length; j++) {
    int end = j;
```

Ternary // -> `currsum = start == 0 ? prefix[end] : prefix[end] - prefix[start-1]`

```
    if (maxsum < currsum) {
        maxsum = currsum;
    }
    sysO ("maxsum = " + maxsum);
}
psvm { int number [] = {1,-1,2,6,-3};
    maxSubarraySum (numbers);
}
```

Max Subarray sum

KADANE'S Problem

Algorithm

+ve + +ve → +ve

+ve + −ve → +ve
small

+ve + −ve → −ve
big

or:

-2, -3, 4, -1, -2, 1, 5, -3

	-2	-3	4	-1	-2	1	5	-3
CS	0	0	4	3	1	2	7	4
MS	0	0	4	4	4	4	7	7

if current sum is having -ve value then update current sum as 0

if -ve value then update 0

-ve → 0

only one for loop is used

Time Complexity = O(n)

```
public static void kadans(int number[])
    {
    int maxsum = Integer.MIN_VALUE;
    int currsum = 0;
for(int i=0; i<number.length; i++) {
    currsum = cs + number[i];
    if (currsum<0) {
        currsum = 0;
    }
    maxsum = Math.max(currsum, maxsum);
}
sysO("outr max subary sum is " + maxsum);

public static v m S A
    {
    int numbr[] = {-2,-3,4,-1,-2,1,5,-3};
    kadanes(numbr);
    }
```

-1	-2	-3	-4	→ Ans = -1

→ smallest -ve element. if all are -ve elements.

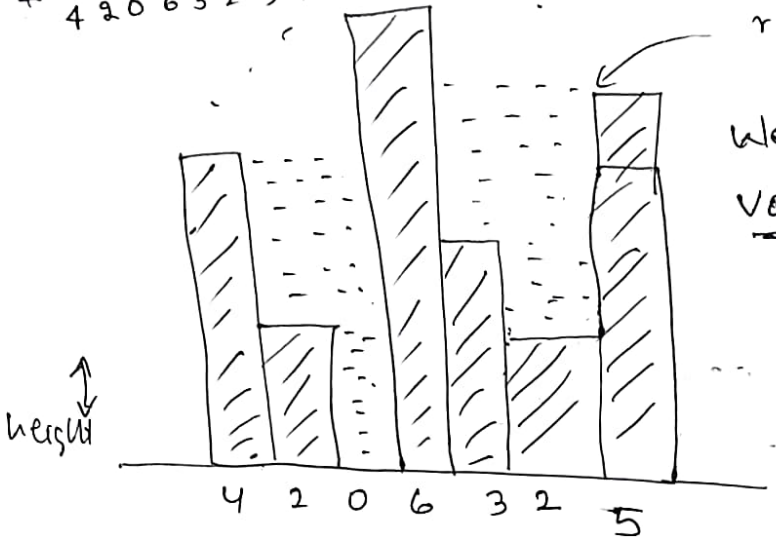
```
currsum = 0;
smallest = integer.MAX_VALUE;
if (currsum[i] < 0)
    {
    currsum = smallest
    }
return smallest;
```

Trapping Rainwater :- (DSA #22) (Medium)

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

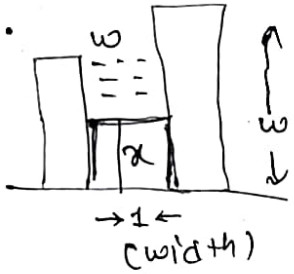
→ Auxillary Arrays

height = [4, 2, 0, 6, 3, 2, 5]



We need to calculate the Volume of rain water

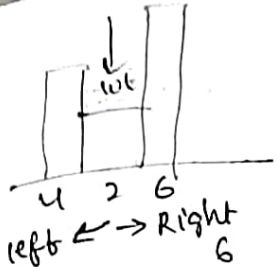
→



(w - x) * 1 = area where pani stored

(water level - bar level) * width = trapped water

iii)



minimum = 4

h = 4 middle bar = 2

wt = (4 − 2) × width
= 1
= 2

(wL − x) × 1
(4 − 2) × 1
= 2

(IV)

min = 6

(wL − x) × 1
6 − 2 = 4

wt = 4
trapped water quantity
= 4

(V)

10 ← 6 2 → 8
10 8 8.

minim = 8

(w − x)
8 − 2
= 6

→ (max left , max Right)
minimum.

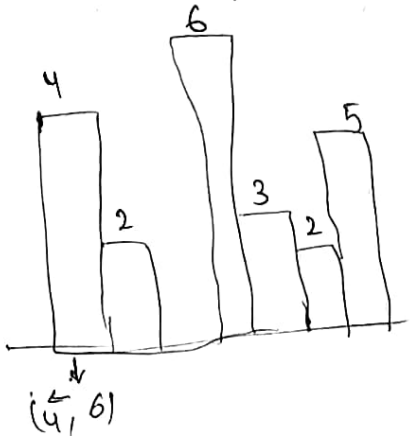
Trapped water = (water level − height) × width (width = 1)

↓

min (maxleft boundary , maxright boundary)

min value is water level

Trapped water = (min − height)



Left right

4 ⇒ (4, 6) min = 4 − 4 = 0

2 ⇒ (4, 6) min = 4 − 2 = 2

0 → 4, 6 = 4 − 0 = 4

6 → 4, 5 = 4 − 6 = −2 = 0

3 → 6, 5 = 5 − 3 = 2

2 → 6, 5 → 5 − 2 = 3

5 → 6, 5 → 5 − 5 = 0

= 0 + 2 + 4 + 0 + 2 + 3 + 0

= 11

(Ans)

Auxiliary Arrays used to find ...
Helper arrays

left max boundary : [| | | | | |]

Right max boundary : [| | | | | |]

minimum value − height

```
public static int trappedRainwater (int height[]) {
    // calculate left max boundary - array
    // calculate r
    int leftmax[] = new int [height.length];
    leftmax[0] = height[0];
    for (int i=1; i< height.length; i++) {
        leftmax[i] = Math.max(height[i], leftmax[i-1]
    }
    // calculate rightmax boundary - array.
    int rightmax[] = new int [height.length];
    rightmax [height.length-1] = height [height.length-1];
    for (int i = height.length-2; i>=0; i--) {
        rightmax[i] = Math.max(height[i], rightmax[i+1]);
    }
    int trapeedwaters = 0;
    // loop
    for (int i=0; i< height.length; i++)
        // waterlevel = min(leftmaxbound, rightboundmax)
    int waterlevel = Math.min(leftMax[i], rightmax[i])
        // trappedwater = waterlevel - height[i];
        trappedwater += waterlevel - height[i];
    }
    return trappedwaters;
```

public static void main (String args[])

{

int height [] = {4, 2, 0, 6, 3, 2, 5};

sysо (trapped Rainwater (height));

}

Time complexity = O(n)

Buy & sell stocks

profit = Buy (Less money) ↓ sell (more money) ↑

profit = selling price − Buying price

you are given an array prices where price[i] is the price of a given stock on the ith day. you want to maximise your profit by chousing a single day to buy one stock and choosing diffesent day in the futur to sell the stock. return the maximum profit you can achieve from this transaction. if you cannot achieve any profit, return 0;

profit = Buy − sey

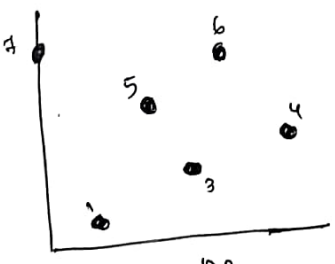
prices = [7, 1, 5, 3, 6, 4]

if 4 3 2 1 0 → decreasing order then return 0. (no propit) so return 0:

Time complexity :-

O(n)

=



Day 1 Bprice=7 x

D2 Sp=1 Buy=7 p = sp − BP 1−7 = −6

D3 Sp=5 BP=1 p=5−1 =4

Day 4 Sp=3 BP=1 3−1 p=2 profit=2

Day=5 Sp=6 Bp=1 profit = 6−1 =5

Day6 Sp=4 bp=1 p=4−1 =3

profit = selling price - buy price

var buy price = +∞.
track, lowest buying price.

var max-valo = 0;

for (int i = 0 ; i = n)
{
(buy price < seling price) - case - 1
{
p = sp - BP
buying price = seling price

Code :-

```
Public static int Buysell stock ( int price[]) {
    int buyprice = Integer.MAX_VALUE;
    int maxprofit = 0;
for (int i = 0 ; i < price.length ; i++)
{
if (buyprice < price[i] & ((Profit-
    int profit = price[i] - buyprice ; // todays profi
                 (selling price)

    maxprofit = Math.max( maxprofit , profit);
}
else {
    buy price = prices[i];
}
}
return maxprofit;
}
ps vm sA ()
{
int prices [] = { 7, 1, 5, 3, 6, 4};
sys o ( byseu stock ( price)) ;
```

return

practice assignment questions for Array (given in PDF)

Basic Algorithms :-

1) Bubble sort :-

{ 5, 4, 1, 3, 2 } unsorted array.

{ 1, 2, 3, 4, 5 } increasing order

{ 5, 4, 3, 2, 1 } decreasing sort.

Inspiration :-

idea :-
large elements come to the end of array by swapping with different elements.

Bubble sort :-

5, 4, 1, 3, 2
swap
4 5 1 3 2
4 1 5 3 2
4 1 3 5 2
4 1 3 2 5

0th turn

n = 5 → 0 to n-2

4 1 3 2 5
1 4 3 2 5
1 3 4 2 5
1 3 2 4 5
1 3 2 4 5 → already.

1st turn

4 terms = (n-1)

0 to n-3

1 3 2 4 5
1 3 2 4 5
1 2 3 4 5 → already done.

2nd turn

0 to n-4

1, 2, 3, 4, 5
1 2 3 4 5
x → already sorted

sorted array.

3rd turn

0 to n-5

for (turns = 0 to n-2) (1st turn to 4th turn)
{ // swap
for (int j = 0; to n-2-turns)

turn = 0 to (5-2-0) ⇒ 3

turn 1 j = 0 to (5-2-1) = 2

code :- bubble sort

```
public static void bubblesort (int arr[]) {
    for (int turn=0; turn < arr.length-1; turn++) {
        for (int j=0; j < arr.length-1-turns; j++) {
            // swap.
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// print array.
public static void printarray (int arr[]) {
    for (int i=0; i < arr.length; i++) {
        Sysoln ("sorted array" + arr[i] + " ");
    }
}

Syso pln();

psvm(sA[])
{
    int arr[] = {5, 4, 1, 3, 2};
    bubblesort (arr);
    printarray (arr);
}
```

swaps = 0
for
for {
swap;
swap++;
if (swap==0)
the
syso (sorted Array)

Time complexity :-

for (t++) n
for (n-1-turns))

Time complexity = O(n^2);

best case -
Time complexity
= O(n)

Selection sort :-

Idea:
pick the smallest (from unsorted) put it at the beginning.

n = 5

5 4 1 3 2

smallest = 1 ⟶

for (int i=0 ; i < n-2)
    smallest = MAX
    for (j = i+1; to n-1)

Time complexity = $O(n^2)$

sorted | unsorted
[5 4 3 2] sm = 2

1 2 | [5 4 3] - smaw
sorted | unsorted

1 2 3 | [5 4] smaw
sorted

1 2 3 4 | [5]
sorted.

code :-

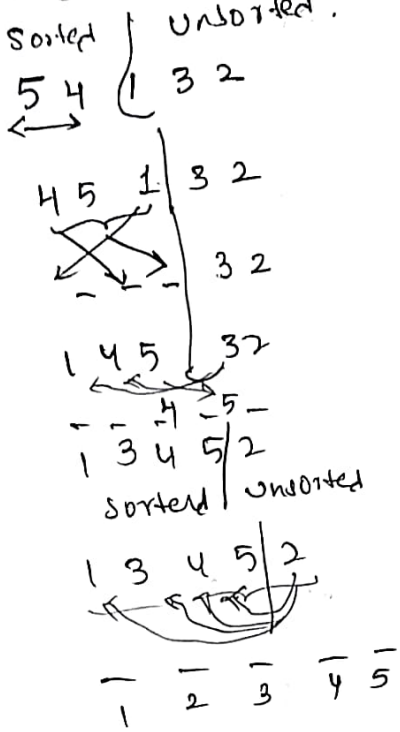
```
public static void selection sort (int arr[]) {
    int n=arr
    for (int i=0; i < arr.length-1; i++) {
        int minposition = i;
        for (int j=i+1; j < arr.length; j++) {
            if (arr[minpos] > arr[j]) {   // < (decreasing order)
                minpos = j;                // > (increasing order)
            }
        }
        // swap
        int temp = arr[minpos];
        arr[minpos] = arr[i];
        arr[i] = temp;
    }
}
public static void main (String args[])
{
    int arr[] = {5, 2, 3, 1, 4}
    selectionSort(Arr);
    for (int i=0; i < arr.length ) i++)
        System (arr[i]);
```

Insertion sort :-

inspiration
↓
cards

idea.

pick an element (from unsorted part) &
place it in the right position (in sorted part)



Time complexity
$= n^2$

selection
bubble
insertion
} n^2

code :-

```
Public static void insertion sort (int a...
    {
    for (int i = 1; i < arr.length ; i++)
        {
        int curr = i ; // arr[i]
        int previous = i-1;
        // finding out correct posion to insert
        while ( prev >= 0 && arr[prev] > arr[curr] )
                                        or (i)
            {
            arr[prev+1] = arr[prev];
            prev--;
            }
        // insertion.
        arr[prev+1] = arr[curr];
        }
    psvm sA ( )
        {
        insertionsort.(arr);
        for (i=0; i<arr.length ; i++)
            { sysO(arr[i]); }
        }
```

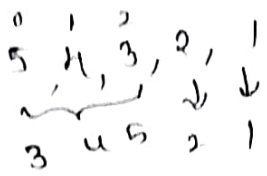
```
public static void insertion (int arr[])
{
for (int i=1 ; i< arr.length; i++)
  {
  int curr = arr[i]
   int prev = i-1;
while (prev >=0 && arr[prev] > curr) {
        arr [prev+1] = arr[prev];
            prev--;
    }
    //insertion;
  arr[prev+1] = curr;
    psvm()
```

Inbuilt sorts :-

```
Import java.util.Arrays;
    Arrays.sort(Arr);
   (class)
TC → O(n log n)

Arrays.sort(arr, si, ei)
 import java.util.Arrays;
 public static void {
     int arr[] = { 2,4,5,6 };
     arr sort
   Arrays.sort(arr);
 //sysoprint(arr[i])
     for(int i=0; i<arr.length; i++)
     { sysO(arr[i] + " ")
       }
```

Arrays.sort(array, starting index, ending index)

Ascending order:-



Starting index = for 0 = 0
1 = 1

Ending index for 1 = 2
2 = 3
3 = 4

ex:- int arr[] = {5,4,1,3,2};

Arrays.sort(arr, 0, 3);
SI EI
0 3

for { sysO(arr[i]); }

For descending order:-

→ Import java.util.collections; // Library

Syntax = Arrays.sort(arr, collections.reverseOrder())

Arrays.sort(arr, si, ei, collections.reverseOrder())

ex:-

```
public static void printArray ( Integer arr[] )
    {
    for (int i=0; i< arr.len ; i++) {
        sysO (arr[i] + " " );
    }
public static void m(SA[])
    {
Integer. arr[] = {5,4,1,3,2};
Arrays.sort (arr, collections.reverseOrder());
Arrays.sort (arr, si, ei, collections.reverseOrder());
  printArray (arr); //function call
```

For collections → (small int) is not possible so for declearing Array larger Integer (Integer) is used ex:-

Integer Arr[] = {1,2,3}

Integer is a primitive type.
reverse order() → works on objects.

counting sort

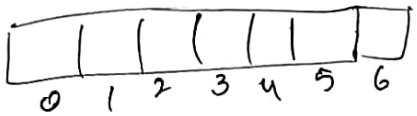
1, 4, 1, 3, 2, 4, 3, 7

min — max — range

= o n. (time complexity)

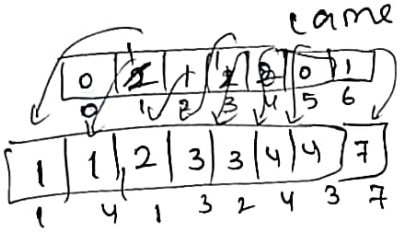
min = 1
max = 7 } range = 7

create count array



store number frequency
(which number
how many times
came)

original



① for (int i=0 to r)
frequency)

② for (int i=0 to max no (range)
range.

(used when range is small)

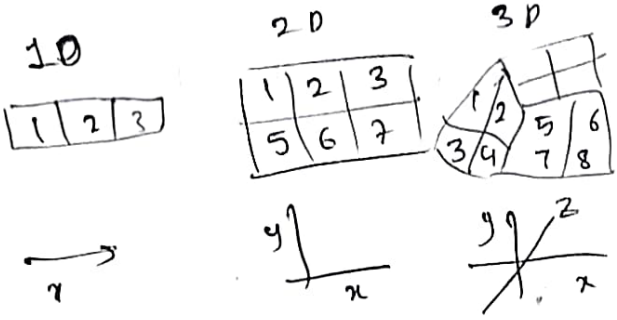
code:-

```
public static void countingSort(int arr[]) {
    int largest = Integer.MIN_VALUE;
    for (int i=0; i<arr.length; i++) {
        largest = Math.max(largest, arr[i]);
    }
    int count[] = new int[largest+1];
    for (int i=0; i<arr.length; i++)
        count[arr[i]]++;
    }
    // sorting. int j=0;
    for (int i=0; i<count.length; i++) {
        while(count[i]>0) {
            arr[j] = i;
            j++;
            count[i]--;
        }
    }
    PSVMSA { function call);
```

8D-Arrays (placement)

Matrix (Rows & columns)

4D ... nD Arrays
(Used in machine Learning)

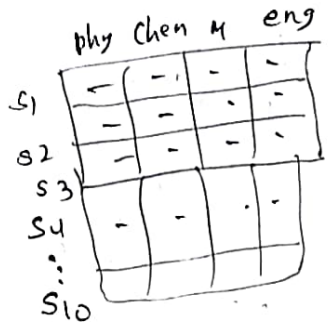


Real life examples :-

S_1 → marks = | 94 | 98 | 99 | 29 |
0 1 2 3

S_2 → ...

S_{10} → ...



RGB :- Cube.

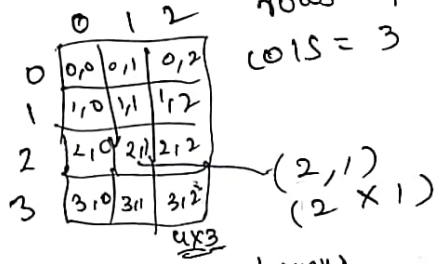
red. Green Blue

(255) (255) (255) white - pixel

(0) (0) (0) black

(0 . . 255) Blue

Representation :-



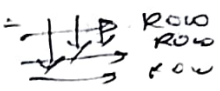
rows = 4.
cols = 3

row :-
for (int i=0 to n)
cols = for (int j=0; m)

Vertical = column
Horizontal = row

n x m
rows cols

2D Array creation :-



psvmsA

```
int matrix[][] = new int [3][3]; // 3x3 matrix
Scanner sc = new Scanner (System.in);
int n = 3, m = 3;   n = matrix.length, m = matrix[0].length.
for (int i = 0; i<m; i++ {
    for (int j = 0; j<m; j++) {
    matrix [i][j] = sc.nextInt()
    }
}
```

output:-

same loop!

```
for (int i = 0; i<n; i++) {
    for (int j = 0; j<m; j++) {
        syso (matrix[i][j] + " ")
    }
    syso.println();
}
```

```
public static boolean search (int matrix[][], int key.
for (int i = 0; i<n; j++) {
    for (int j = 0; j<m; j++) {
    if (matrix[i][j] == key) {
        syso ("found at " + i + j;
        return true;
    }
}
return false;
```

psvmsA

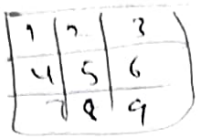
```
search (int matrix);
```

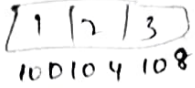
2D in memory:

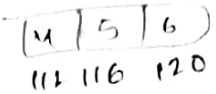
int i[3][3]

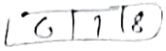
row major :-

rows →



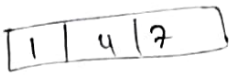






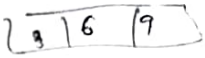
100 104 108 112 116 120

column major :-



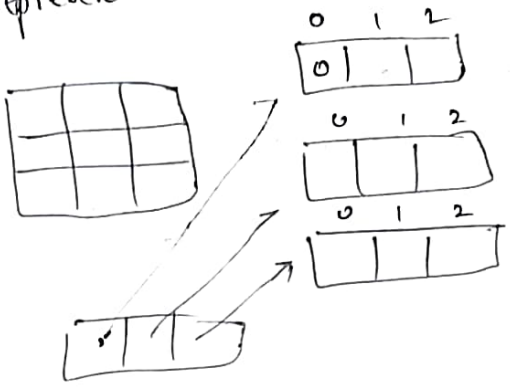
c1

c2



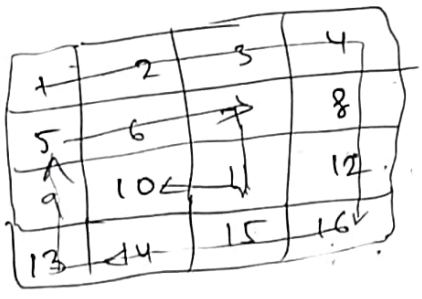
c3

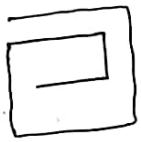
representation.



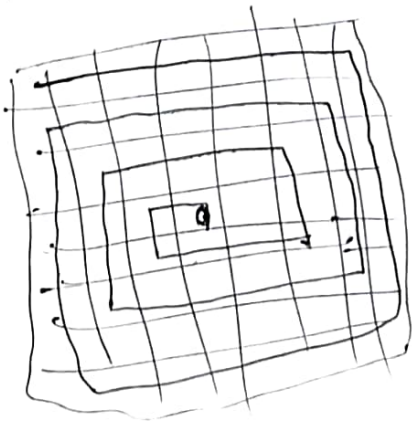
Spiral Matrix (Amazon Apple google microsoft oracle)

print in spiral manner





output = 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10

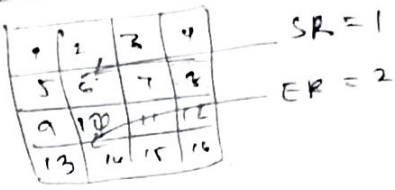


related to border

Approach :- Border

→ for each one → it goes to
1) 2 rows
2) 2 columns.

→ startROW
→ end row
→ start col
→ end col



ER = n - 1 = 3

	1st iteration	2nd	
start row	0	1	++
end row	3 (n-1)	2	--
start col	0	1	++
end col	3 (m-1)	2	--

Update
start R = ++
end row = --
start col = ++
end col = --

```
while ( )
{
    1) top boundary
    2) Right part
    3) bottom part
    4) left part
```

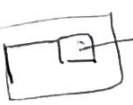
· SR ✓ Top
SC → EC
EC ✓ right
SR+1 → ER

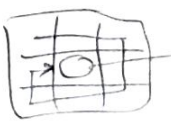
ER ✓ bottom.
EC-1 → SC
SR ✓ left.
ER-1 → SR+1

nxn n=4

n = 3, 5, 7 odd
SR ≤ ER && SC ≤ EC

nxm = odd ( )

if  → start = end
break



Code :-

```
public static void printSpiral(int matrix[][]) {
    int startRow = 0;
    int startCol = 0;
    int endRow = matrix.length - 1;
    int endCol = matrix[0].length - 1;

    while (startRow <= endRow && startCol <= endCol) {
        // top → printing columns
        for (int j = startCol; j <= endCol; j++) {
            System.out.print(matrix[startRow][j] + " ");
        }
        // right ↓ printing rows
        for (int i = startRow+1; i <= endRow; i++) {
            sysO(matrix[i][endCol] + " ");
        }
        // bottom
        for (int j = endCol-1; j >= startCol; j--) {
            if (startRow == endRow) {
                break;
            }
            sysO.pt(matrix[endRow][j] + " ");
        }
        // Left
        for (int i = endRow-1; i >= startRow+1; i--) {
            if (startCol == endCol) {
                break;
            }
            sysO(matrix[i][startCol] + " ")
        }
        startCol++;
        startRow++;
        endCol--;
        endRow--;
    }
    sysO(pln();
```

```
public static void main (String args[]) {
    int matrix[][] = { {1,2,3,4},
                       {5,6,7,8},
                       {9,10,11,12},
                       {13,14,15,16} };

    printSpiral (matrix); // fn call
}
```

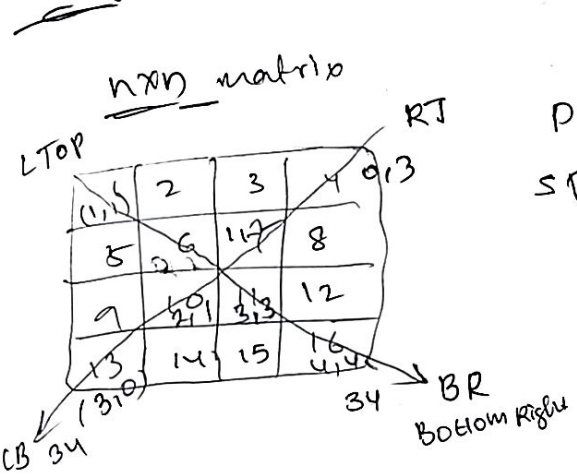
op = 1 2 3 4 8 12 16 15 14 13 9 6 7 11 10.

Diagonal Sum

Samsung
Amazon
Microsoft

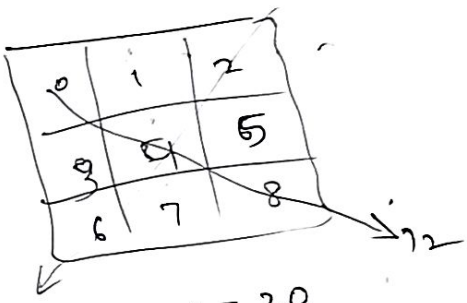
Time complexity = $O(n^2)$

nxn matrix



PD = 1+6+11+16

SD = 4+7+10+13



PS = 20

n = even

⇒ (i = j) (row = col) sum. (primary diagonal ↘

⇒ i + j = n-1 (secondary diagonal) ↙

Code:

```
public static int diagSum (int m[][]) {
    int sum = 0;
    for (int i=0; i<matrix.length; i++) {
        for (int j=0; j<m[0].len; j++) {
            if (i==j) {
                sum += matrix[i][j];
            }
        }
    }
```

```
    else if (i+j == matrix.length-1) {
        sum += matrix[i][j];
    }
}
return sum;
```

```
psvm s[] {
    int a[][] = {
    // function call
```

Optimal solution (T.C = O(n))

primary diagonal = (i = j)(i, i)

secondary diagonal = (i + j = n - 1)

(j = n - 1 - i)

[time complexity = O(n)]

```
p static diagsum (int m[][])
    {
    int sum = 0;
for (int i=0; i<matrix.length; i++) {
    // primary diagonal
    sum += matrix[i][i];
    // secondary diagonal if (i != matrix.length-1-i) {
    sum += matrix[i][matrix.length-i-1]
                    (matrix.length-1-i)
        }
    return sum;
psvm s+()
    {
    int arrmatrix[][] = { 1 2 3
                          4 5 6
                          7 8 9 }
    diagsum (matrix) // call
```

Search in Sorted Matrix (Adobe, oracle)

sorted.

ascending →

10	20	30	40
15	25	35	45
27	29	37	48
32	33	39	50

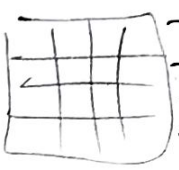
Ascending sorted ↓

Search for a key in row wise & column wise sorted matrix.

Approach :-

1) → Brute force - Attack. → row wise
worst case = $O(n^2)$ → col wise.

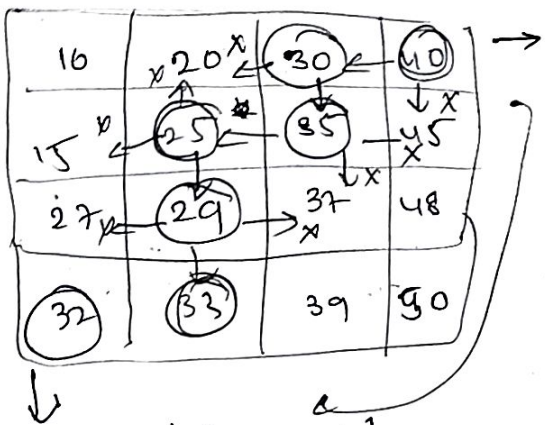
2) → Row wise Binary search $O(n \log n)$



$= n(\log n)$
$O(n(\log n))$

3) Staircase search.

key = 33



(0, m-1)
key < cell value
Left
key > cell value
bottom.

(n-1, 0)
key < cell value
Top
key > cell value
Right

(0, m-1)
key < cell value
Left
key > cell value
Bottom.

for i=0 to n-1
for j=m-1 to 0

Code !!!)

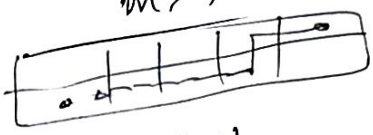
```
Public s boolean staircasesearch (int matrix[][], int k)
{
int row=0', col= mat[0].length -1;
while(row< matrix.length && col >=0) {
if ( matrix[row][col] ==key) {
        syso ("found at " + row, + col);
return true
}
else if (key < matrix[row][col]) {
        col--; // push left
```

```
else {
    row++ ; pushdown
}
}
return false // if not found.
psvm s+(n) {
  in. matrix[][] = {{ }
                   { }
                   { }
                   { }
                   }

  int key = 100;
  staircase search (matrix, key);
}
```

Time complexity =

worst case = O(n)

n>m



m>n

O(m)

Time complexity of staircase search = O(n+m)

Strings :-

which is used to store the character sequence of characters.

String str = "abcd";

String capital.

char arr[] = {'a', 'b', 'c', 'd'};

string str = "abcd"; // 1st process

string str2 = new string ("xyz"); // 2nd method of creation.

// strings are Immutable. Very very Important.

strings will not change

to change we need to create a new string.

strings Input :-

```
scanner sc = new scanner (sys.in);
string name = sc.next(); //nextLine();
sysо (name);
```

i/p :- name - TOny

o/p name Tony

Spaces are not allowed

Length of String = string name.length();

variable.length()

```
String name = "sai";
syso (name.length);
```

Concatenation :- Adding string 1 & string 2.

eg:
```
String first_name = "sai";
String last_name = "kiran";
String name = first_name + "_" + last_name;
Syso ("name");
```

• charAt( ); to print the indent at which one

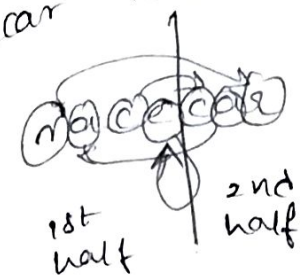
```
name.charAt(0);

public static void print( String str){
  for(int i=0; i< str.length(); i++) {
    syso (" str. charAt (i));
  }
  syso.pln()
}

psvms A {
  String fullname = "saikiran";
  // call print (fullname);
```

f_name = "1234";
Syso print char → it gives capital or random characters

Q-1 Multi company asked

check if a string is a palindrome or not

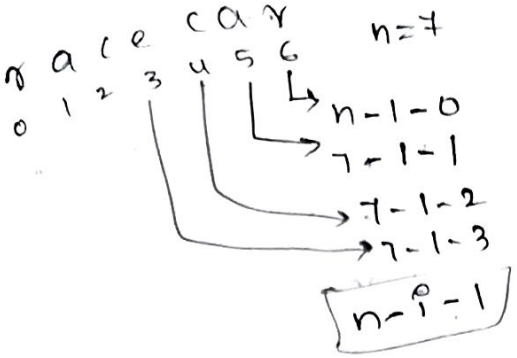
racecar noon madam.



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

mid.

string[i] == string(n-i-1) or (n-1-i)



Time complexity

= O(n)

Code:-

```
public static boolean ispalindrome(String str)
{
for(int i=0; i<str.length/2; i++)
{ int n=str.length();
  if(str.charAt(i)!=str.charAt(n-1-i))
  {
    return false;
  }
}
return true;

psvm(String args[]){
  String str="madam"
  Syso print(ispalindrome(str));
```

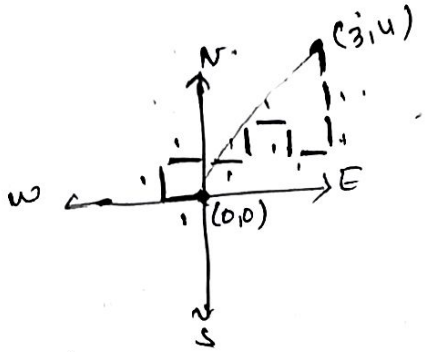
if elements(i) & (n-i-1) are not equal it is not palindrome

if the above condition is continued until end then it will return true.

Question 2:-

Given a route containing 4 directions (E, W, N, S) (East, West, North, South).
Find the shortest path to reach destination.

"NONENESENNN"



shortest distance $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

$(x_1, y_1) = (0,0)$ $(y, 42) = (3,4)$

$= \sqrt{(3-0)^2 + (4-0)^2}$

$= \sqrt{9+16}$

$= \sqrt{25} = 5$

Time complexity $= O(n)$

$x = 0$	N ↑ → $y+1$	Logic
$y = 0$	S ↓ − $y-1$	
(origin)	W ← $x-1$	
default.	E → $x+1$	

Code :-

```
public static float shortestdistance(String path) {
    int x=0 , y=0 ;
    for (int i=0 ; path.length ; i++)
        {
        char direction = path.charAt(i);
        // south
        if (dir == 'S') {
            y--;
        // north
        else if ( dir == 'N') {
            y++;
        // west
        else if (dir == 'W') {
            x--
        // east
        else {
            x++
```

return int x^2 = (x x x) // $(x_2-0)^2 = x^2$

int y^2 = (y x y) // $(y_2-y_1)^2 = (y_2-0)^2 = y_2^2 = y^2$

return (float) Math.sqrt($x^2 + y^2$) // convert into float because by default math gives boolean value

psvmSA() {

String path = "WNEENESNNN";

Sysо(shortestdistance(path));

String s_1 = "TONY"

String s_2 = new String ("TONY");

if ($s_1 == s_2$)

Sysо("same");

else

Sysо("notsame");

Here String s_1 = "TONY" ≠ new String ("TONY");

↓ ↓

simply create a TONY ≠ "TONY" New keyword create a new memory to "TONY"

• equals

s_1.equals(s_3) ⇒ $s_1 == s_3$

returns boolean value.

we use • equals

Substring :-

Substring

Hello world → low (substring)

Hwd → subsequence.

0 to 4 means 0, 1, 2, 3 | last ending index is not printed.
not included

3, 5 = 3, 4

```
Public static String substring (String str, int si, int ei)
    {
    String substr = " ";
    for (int i=si; i<ei; i++) {
            substr = substr + str.charAt(i);
            }
        return substr;
    }
p svm s {
String str = "Helloworld";
    sys o. (println (substring (str, 0, 5));
    }
}
```

o/p = hello.

(or) Java contains inbuit Substring( )

```
String str = "Helloworld");
sys o( str.substring(0,5);
```

// Hell (Same opt)

Question 2:-
for a given a set of strings, print the largest string.

Lexicographic

aaabc (CLd)
aaabd
Largest

Apple Banana.

a is less than b

compareTo → used to compare 2 strings

str1.compareTo(str2)

↓

if
0 : equal
< 0 : -ve
> 0 : +ve

str1 < str2
str1 > str2

Time Complexity
=
O(x x n)
↓

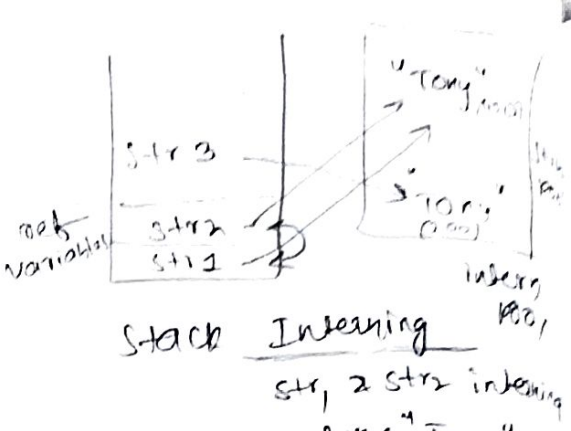
Ignore case compare to.

compareToIgnoreCase → A & a are equal
A == a

```
public static void main (String args[]){
String fruits[]= {"apple", "mango", "banana"};
   String largest = fruit[0]; // first string = 0
for(int i=1; i<fruits.length; i++){
  if (largest.compare(fruits[i]<0){
          largest = fruits[i];
  }
  sysO(largest);
```

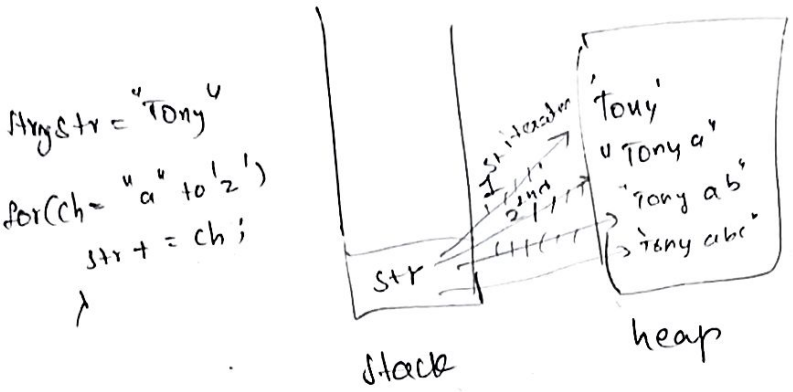
What are Strings:-

String str1 = "Tony"
String str2 = "Tony"
String str3 = new String ("Tony");
↓
Interning.



Stack Interning

str1 & str2 interning same "Tony"



→ Tony a
- Tony ab
- Tony abc

Time Complexity

$= O(n \times m)$ (or)
$O(n^2)$

StringBuilder :-

StringBuilder is a datastructure in which updations are happened in same string.

but in String updations are happened in another stored string.

Syntax:

```
StringBuilder sb = new StringBuilder("Hello");
sb.toString();
```

toString → convert any object to String

int a = 10; ✗
a.toString()
not possible

Integer a = 10; ✓
a.toString();
possible

char ch = 'a'
ch.toString ✗

character ch = 'a'
ch.toString()

Time Complexity = O(26)
for number = O(26 * n²)

coder:

```
public static void main (String[]) {
    StringBuilder sb = new StringBuilder ("");
    for (char ch = 'a'; ch <= 'z'; ch++) { // append need to add before.
        sb.append(ch);
    }
    sysO (sb);
```

Question 4 (Codenation company)

For a given string convert each the first letter of each word to upper case.

"hello world"
Hello World;

"hi, iam shradha"

o/p = Hi I Am Shraddha.

word come after empty space.

Characters.toUpperCase(ch);

'a'

ch = 'A'

```
Public static String toUpperCase(String str) {
    StringBuilder sb = new StringBuilder("");

    char ch = Character.toUpperCase(str.charAt(0));
    sb.append(ch);

    for(int i=1; i<str.length(); i++) {
        if(str.charAt(i)=='_' && i<str.length()-1)
            sb.append(str.charAt(i));
            i++;
            sb.append(Character.toUpperCase(str.charAt(i)));

        else {
            sb.append(str.charAt(i));
    return sb.toString();

psvms A {
    String str = "hi, iam, sai";
        syso(toUpperCase(str));
```

Question -5

String Compression (Amazon)

"aaabbcccdd" → a3b2c3d2

"aaabbbdd" → a4b3d2

"abc" → abc → less strength

a1b1c1 → more strength ✗

Approach:

count

for (int i=0 to str.length)

ch(i)

count=1

while (repeat) {

count++

i++

}

count > 1 — number

=1 → x

Time complexity

$= O(n^2)$

i++ } 2 loops for & while.
i++ }

Code:-

```
public static String Compress(String str) {
    String newstr = " ";

    for (int i=0; i< str.length(); i++) {
        Integer count = 1;
        while ( i< str.length()-1 &&
                str.charAt(i) == str.charAt(i+1)
                count++;
                i++;
        }
        newstr += str.charAt(i);
        if (count > 1) {
            newstr += count.toString();
        }
    }

    return newstr;

public SVMS()
{
    Str str = "aabbccdd";
    Syso (Str(compress(str));
```