A LITTLE PROGRESS EACH DAY ADDS UP TO BIG RESULTS

{Arrays, Interview Problems, M-A, B.T, Sorting, Strings, Hashing}

Recursion → [C & O, LL, Trees.]

↓

trailer. → Advanced. → Hero.

Algorithms, technique.

Villian.

Error

[TLE, MLE, Stack overflow,

→ OOPs {x}

→ syntax {search}

→ class & object concept.

Class → It is a blue-print.

Eg → floor plan of an apartment / building / house.

object → Real instance of the class.

Eg → Physical apartment / building / house.

[For one class, we can have multiple objects.]

Class
- Attributes – to define data
- Methods / Functions – to define functionalities.

```
class Car {
    name
    color
    price
    mileage
    :
    drive() { — }
    break() { — }
    A.C() { — }
}
```

Car : Dhiraj
- name → Tesla Mark-5
- color → white
- price → $180K.
- mileage → 400 miles / charge

```
drive() { — }
break() { — }
A.C() { — }
```

Car : Faran
- name → BMW X-5
- color → Black
- price → 50L
- mileage → 3 km/L

```
drive() { — }
break() { — }
A.C() { — }
```

→ [Same functionalities for all the objects]

```
class Student {
    Attributes:
        name
        I.d
        :
    Functionalities:
        study ( ) { - }
        sleep ( ) { - }
        sports ( ) { - }
        Exams ( ) { - }
}
```

Student s1 = new Student()

(#2368)

object reference of student class.

name → "Gaurav"
I.d → "123"
(#2368)
Memory address.

s1.name = "Gaurav"
s1.id = "123"

{ dot is used to access attributes and functionalities }

s1.study()

Student s2 = new Student()

s2.name = "Nitin"
s2.id = 456

name → ~~Nitin~~ John
id → 456
#4597

Student s3 = s2

shallow copy

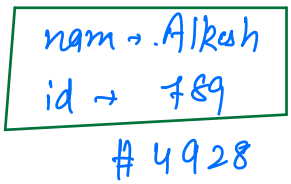
Student s6 ; {null}

print(s6.name)

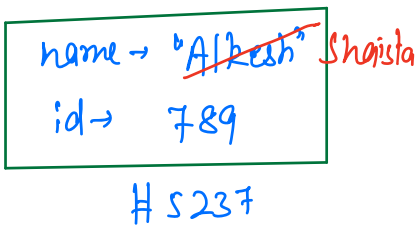
Error!! Null Pointer Exception

print(s3.name) → Nitin
s3.name = John
print(s3.name) → John
print(s2.name) → John.

```
Student s4 = new Student();
s4.name = "Alkesh"
s4.id = 789
```




```
Student s5 = new Student();
s5.name = s4.name;
s5.id = s4.id;
```



#Deep Copy

```
print(s5.name) → Alkesh
s5.name = "Shaista"
print(s5.name) = Shaista
print(s4.name) = Alkesh
```

Q: Create a rectangle class that supports following functionalities

① Find the area of the rectangle.

② Check if the rectangle is square or not.

```
class Rectangle {
    int l, b;
    Rectangle(int x, int y){
        l = x, b = y.
    }
    int area() {
        return l*b
    }
    boolean isSquare() {
        return (l == b)
    }
}
```

```
Rectangle r = new Rectangle();
r.l = 4;
r.b = 6;
```

l → 4
b → 6

2222

magic

```
Rectangle r = new Rectangle(4,6);
```

Constructor

→ method to initialize the attributes of class at the time of object creation.

① name of this method must be exactly similar to the class name.

② No return type. {not even void}

Q: Given N rectangles with length & breadth in A[] and B[].

(A[i], B[i]) → i^{th} - Rectangle.

Find the sum of area of rectangles which are not square.

[using Rectangle class]

A- {2 5 3 6 2}
0 1 2 3 4

B- {4 5 1 6 2}
0 1 2 3 4

{ans = 11}

```
ans = 0
for(i = 0; i < N; i++){
    Rectangle r = new Rectangle(A[i], B[i])
    if( r.isSquare() == false){
        ans += r.area();
    }
}
return ans;
```

```
for(i = 0; i < N; i++){
    if(A[i] != B[i]){
        ans += A[i] * B[i]
    }
}
```

Readable.

Re-usable.

```
ans = 0, Rectangle r;
for(i = 0; i < N; i++){
    r = new Rectangle(A[i], B[i])
    if( r.isSquare() == false){
        ans += r.area();
    }
}
return ans;
```

Q: Add a method / function to chuck if area is

① greater than integer K.

② greater than area of another rectangle.

```
class Rectangle {
    int l, b;
    Rectangle(int x, int y){
        l = x, b = y.
    }
    int area() {
        return l*b
    }
    boolean isSquare() {
        return (l == b)
    }
}
```

this / self ⇒ reference of current calling object

```
boolean isGreaterThan(-K) {
    return this.area() > K;
}
```

Method/Function Overloading

```
boolean isGreaterThan(Rectangle r){
    return this.area > r.area();
}
```

```
Rectangle r1 = new Rectangle(4, 6);
Rectangle r2 = new Rectangle(5, 5);
print(r1.isGreaterThan(20));
print(r1.isGreaterThan(r2));
```

Method overloading.

→ same function name

→ different parameters

Q) Given N rectangles with length & breadth in A[] & B[]. ∀ index i, count the no. of squares on left of i such that their area is greater than area of current rectangle.

```
int[] ans = new int[N];

Rectangle[] arr = new Rectangle[N];
```

Rectangle → custom data-type.

A = [2 5 3 6 2] (0 1 2 3 4)

B = [4 5 1 6 2]

area → 8 25 3 36 4

ans → 0 0 1 0 2

A → [(2,4) (5,5) (3,1) (6,6) (2,2)]

```
for( i = 0; i < N; i++) {
    arr[i] = new Rectangle(A[i], B[i])
}
```

```
for( i = 0; i < N; i++) {
    count = 0
    for( j = 0; j < i; j++){
        if( arr[j].isSquare() && arr[j].isGreaterThan(ans[i])){
            count += 1
        }
    }
    ans[i] = count;
}
return ans[]
```

T.C → O(N²)

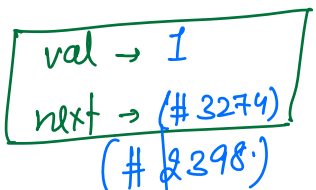
S.C → O(N)

Object Reference Inside a Class

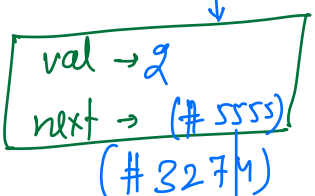
```
class Node {
    int val;
    Node next;
        ↳ object reference
          of node class.

    Node ( int x) {
        val = x
    }
}
```

Node a = new Node(1);



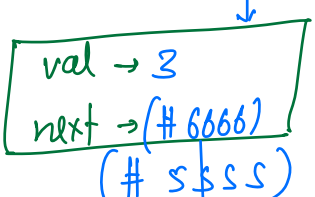
Node b = new Node(2);



a.next = b

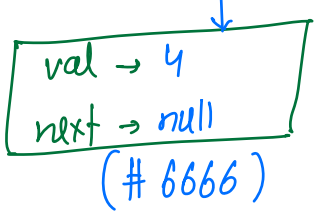
print(a.next.val); ⇒ 2 ✅

Node c = new Node(3);



b.next = c

Node d = new Node(4);



c.next = d

D.S.

Linked-List

{next session}

[Whole movie is pending !!]

Kth symbol →

0 → 01
1 → 10

1 → 0 (1)
2 → 0 1 (2)
3 → 0 1 1 0 (4)
4 → 0 1 1 0 1 0 0 1 (8)
5 → 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 (16)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

A → row no.
B → index.

[Value at B^{th} idx of A^{th} row?]

A → 1st half. | 2nd half. (2^{A-1})

$[0 \rightarrow \frac{2^{A-1}}{2})$

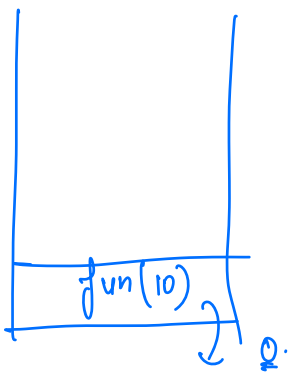
$fun(A-1, B)$, $1 - fun(A-1, B - 2^{A-2})$

↳ check this.

```
function(n){
    if(n%2==0){return 0}
    return function(n-1) + function(Math.floor(n/2));
}
```

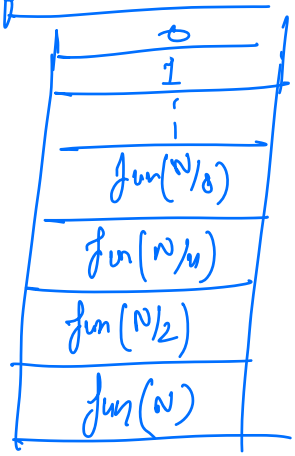
T.C → O(1) | worst case → O(log N)

if n is even. →



T.C → O(1)

if n is odd.



```
int bar ( x , y ){
    if (y == 0) return 0
    return (x + bar(x, y-1))
}
```

```
int foo (x , y) {
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

foo(3,5)

```
int foo (x=3, y=5){
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

↓

```
int foo (x=3, y=4){
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

↓

```
int foo (x=3, y=3){
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

↓ ↑ 9

```
int foo (x=3, y=2){
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

↓ ↑ 3

```
int foo (x=3, y=1){
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

↓ ↑ 1

```
int foo (x=3, y=0 ){
    if (y == 0) return 1
    return bar (x, foo(x, y-1));
}
```

9

```
int bar ( x=3, y=3){
    if (y == 0) return 0
    return (x + bar(x, y-1))
}
```

1 ↓ ↑ 6

```
int bar (x = 3 , y=2) {
    if (y == 0) return 0
    return (x + bar(x, y-1))
}
```

1 ↓ ↑ 5

```
int bar (x = 3 , y=1){
    if (y == 0) return 0
    return (x + bar(x, y-1))
}
```

↓ ↑ 3

```
int bar ( x=3 , y = 1){
    if (y == 0) return 0
```

```
int bar ( x = 3, y=1){
    if (y == 0) return 0
    return (x + bar(x, y-1))
}
```

↓ ↑ 0

```
    return (x + bar(x,y-1))
}
```

↓ 0

```
int bar(x=3, y=0){
    if(y==0) return 0
    return (x + bar(x,y-1))
}
```

```
int bar(x=3, y=0){
    if(y==0) return 0
    return (x + bar(x,y-1))
}
```