Recursion

→ big Problems → small problems

→ function calls itself

key Parts of Recursion

1) Base case - The condition where the function stop calling itself

2) Recursive case - The part where the function keeps calling itself with a smaller input

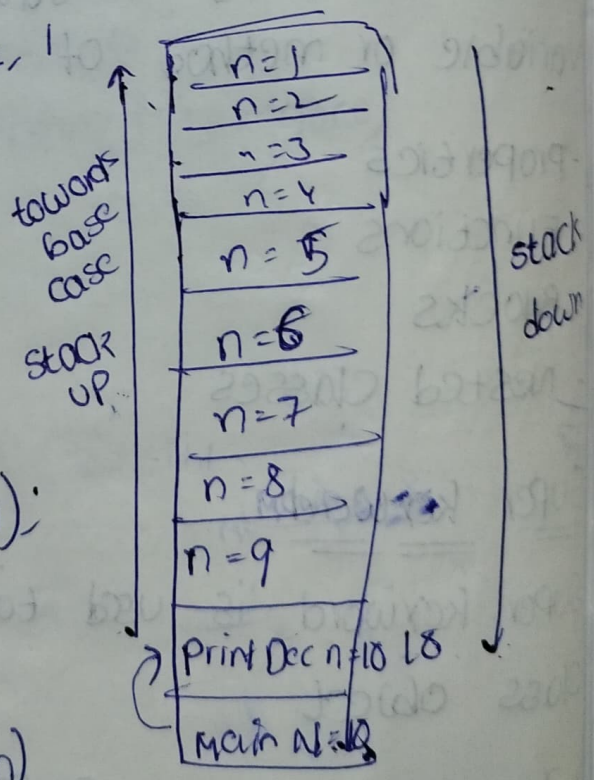
Problems

1) logic Print num from m to 1. (Decreasing order)

n=10

10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Call Stack

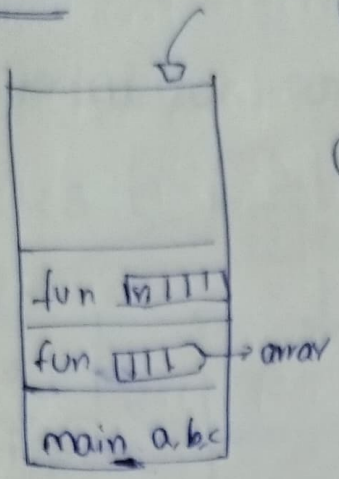


Decreasing order

```
if(n==1) {
    System.out.println(1);
    return;
}
System.out.println(n)
printDec(n-1);
```

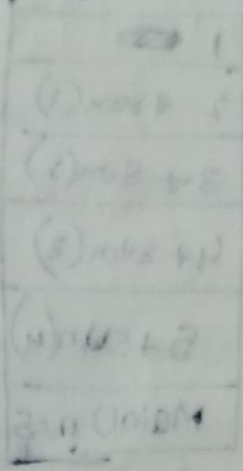
Stack overflow

Base case
↓
not their
means
(stack overflow
occur)



① Parameter memory ↑ → very high

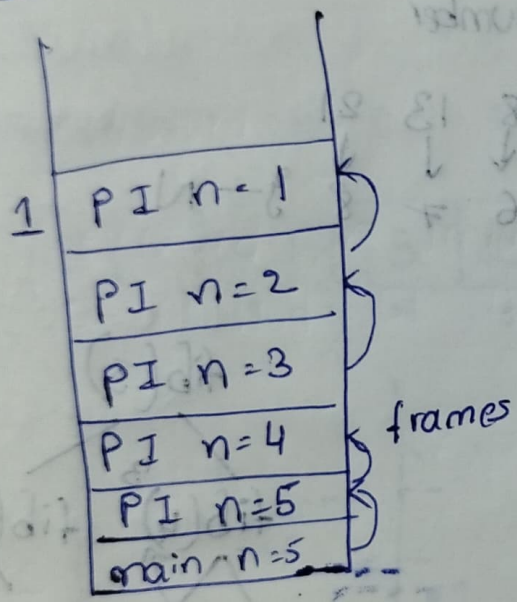
② too many calls



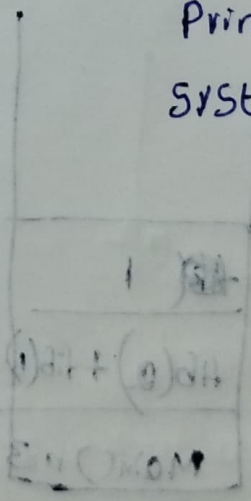
Problem 2

n to 1 (increasing order)

Call Stack

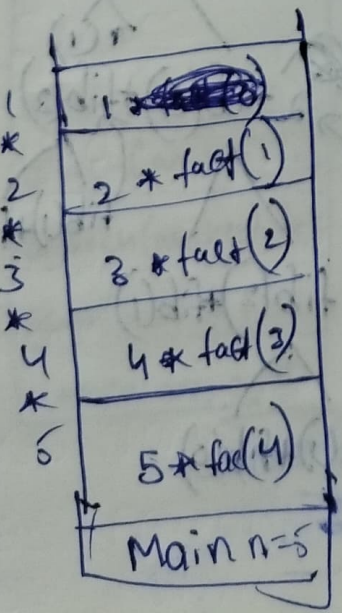



```
Public static void Print(
                int n) {
if (n == 1) {
System.out.Println(1);
return;
}
Print(n-1);
System.out.Println(n);
```



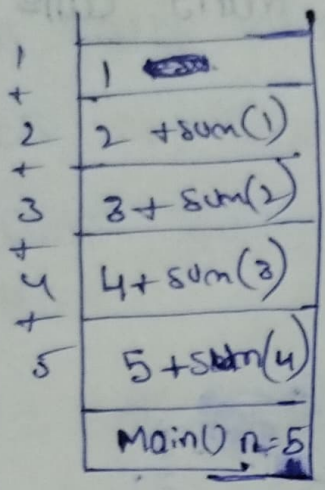
Problem 3

fact n



Problem 4

Sum of n numbers



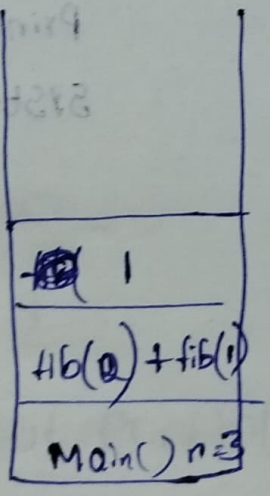
Problem 5

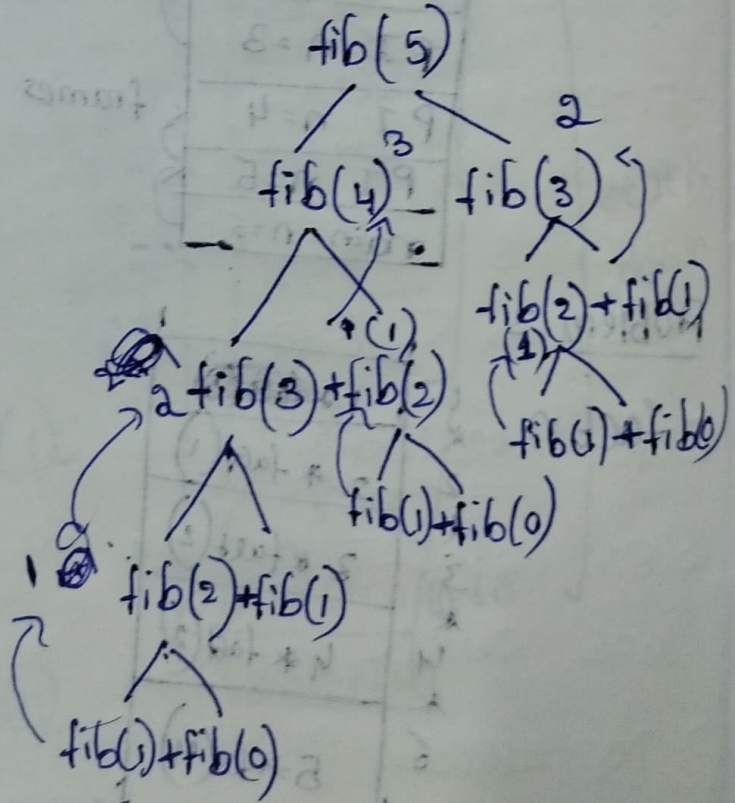
Print N^{th} fibonacci number.

Series → 0 1 1 2 3 5 8 13 21

1 2 3 4 5 6 7 8 ... N

0^{th} 1 2 3 4 5

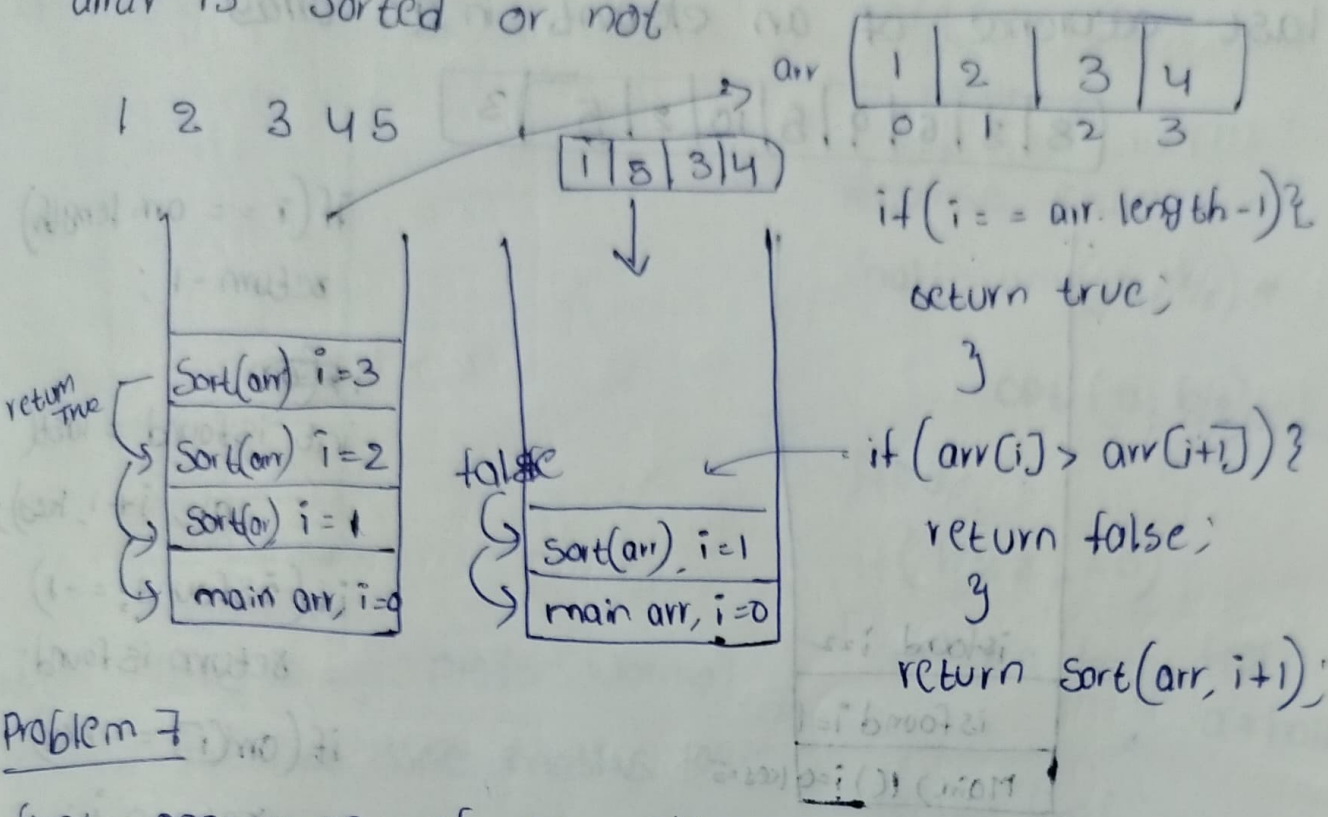




Problem 6

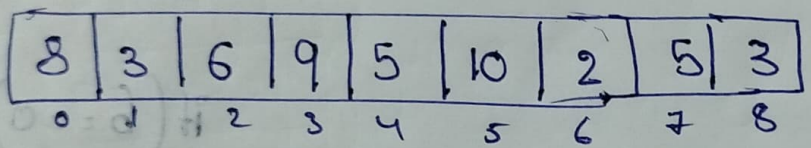
array is sorted or not

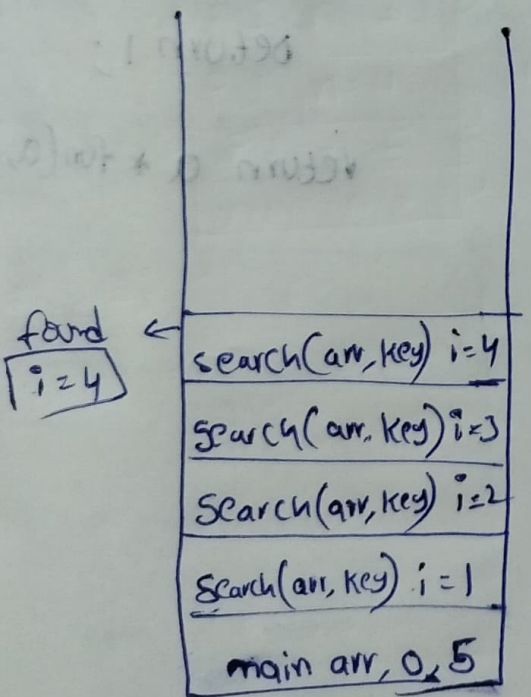
1 2 3 4 5



Problem 7

first occurence of an element





base case

```
if(i==arr.length){
    return -1; }
if(arr[i]==key){
    return i; }
```

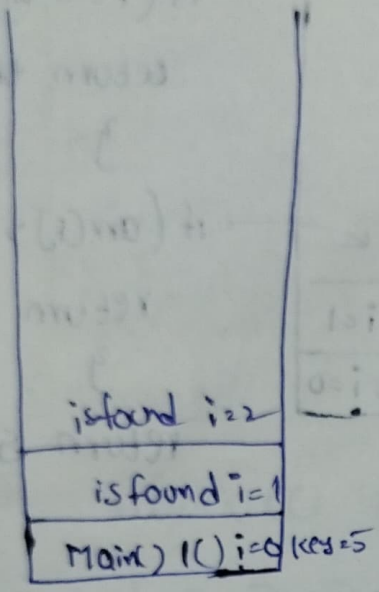
recurse case

search(arr, i+1, key)

Problem 8

last occurence of an element in an array

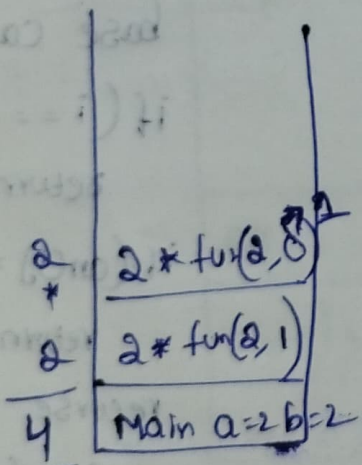
8	3	6	9	5	10	2	5	3




```
if(i == arr.length)
    return -1;
int isfound = last(
        arr, i+1, key);
if(isfound != -1)
    return isfound;
if(arr[i] == key)
    return i;
```

Problem 9

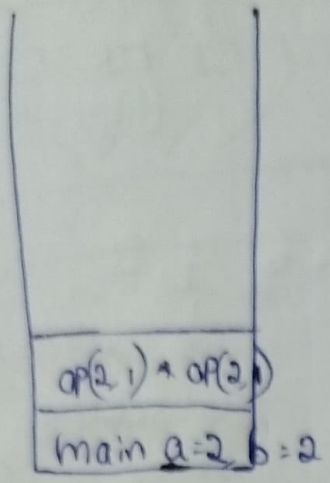
Print x^n




```
if(b == 0)
    return 1;
return a * fun(a,b-1);
```

Problem 10

Optimized Power a^b



base case is

```
if(b==0)
  return 1;
```

recurse case

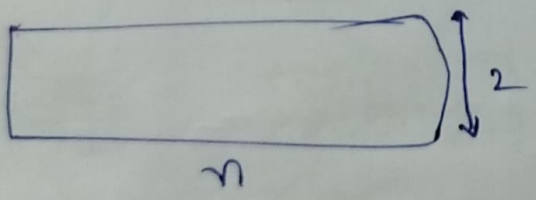
```
half = opt(a, b/2) *
       opt(a, b/2)
//odd
if(b%2!=0)
  return half = half
                a*half;
return half;
```

Note: Mostly we prefer Normal Method or this one maths logic high.

Problem 11

Tiling Problem This is Advanced hard

2 x n → floor



combo

vertical $2*n-1$

horizontal $2*n-2$

base case

```
if(n==0||n==1)
  return 1;
recurse case
h = 2*f(n-1)+
    2*f(n-2)
```

*