# Functions

$f(x) = x+5;$

input = 2 output = 7

$f(a, b) = a + b;$

input = 2, 3 output = 5

---

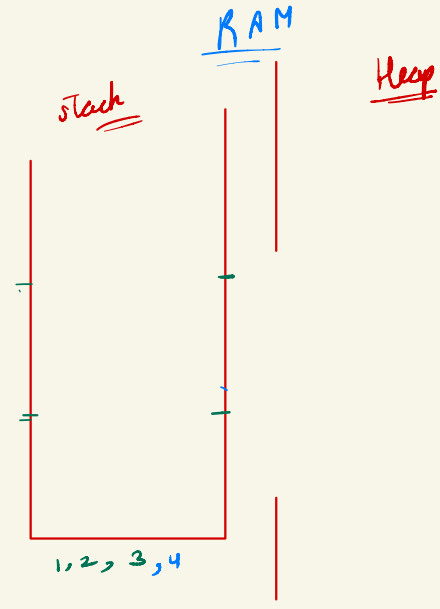
⇒

```
public static int sum(int a, int b){
    int sum = a+b;
    return sum;
}
```

```
public static return_type fun_name (parameters) {
    // logic
}
```

(return_type: void)

```
public class Main {
    public static int sum(int a, int b){
        int sum = a + b;
        return sum;
    }

    public static void main(String[] args) {
        int a=2;
        int b=3;

        int ans=sum(a, b);
        System.out.println(ans);
    }
}
```



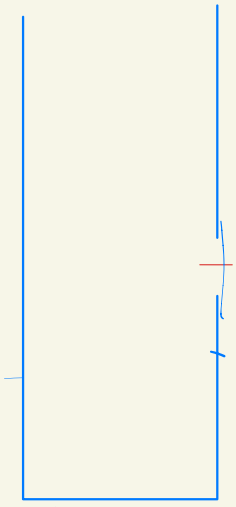

5

```java
public class Main {
    public static void printSomething(){
        System.out.println(x: "Hi!!");
        int a=5;
        a++;
        a--;

        a=11;
    }

    public static int sum(int a, int b){
        printSomething();

        int sum = a + b;
        return sum;
    }

    Run | Debug
    public static void main(String[] args) {
        int a=2;
        int b=3;

        int ans=sum(a, b);
        System.out.println(ans);
    }
}
```

printSomething();

Hi!!

5



# PMI

Principal of Mathematical Induction

sum of first n natural number

$$= \frac{n(n+1)}{2}$$

$n = 0,\ n = 1$ → base case

suppose $n = k$ → smaller problem

↳ $n = k+1$ → solving

# Recursion

a function calling itself

1) solve for the smallest problem
2) keep a faith that it will work for a smaller problem. (faith)
3) solve for rest of the problem. (to reach your expectation)

# print decreasing

1) n=1 is the smallest problem
2) n=5, (n-1)
   4

```
if (n==1)
    Sys out (1);
    return;
```

---

Faith (smaller value)

n-1 ⇒ 4

4 3 2 1

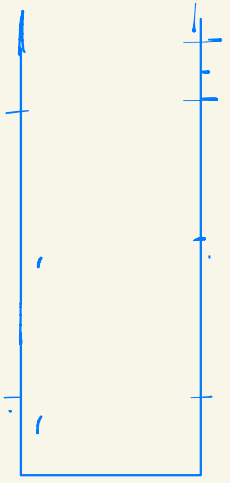
Expectation

If I provide n=5, it will 5 4 3 2 1

very important

10 lines → 2 pages of notebook

```
public static void pd(int n){
    if(n==0){
        return;
    }

    System.out.println(n);
    pd(n-1);
}
```



Ques print increasing recursively.

n=5

1) you won't be able to learn recursion if you don't dry run.

Faith

It will print 1 numbers for (n-1);

4 ⇒ 1 2 3 4

```
public static void PI (int n){
    if(n==0) return;
    1) PI (n-1);
    2) System.out.println(n);
}
```

Expectation

1) my function should print increasing numbers.
2) for n=5, it should print 1 2 3 4 5

```
public static void PI(int n){
    if(n==0){
        return;
    }

    PI(n-1);
    System.out.println(n);
}
```

break till ⇒ 10:15

# print decreasing increasing

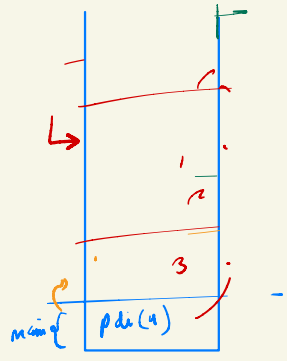
Faith

n-1 n=3

3 2 1 1 2 3

4 3 2 1 1 2 3

```
public static void PDI(int n){
    if(n==0) return;

    1) System.out.println(n);
    2) PDI(n-1);
    3) System.out.println(n);
}
```




Expectation

It will print numbers in decreasing, then increasing.

n=4

4 3 2 1 1 2 3 4

# Find factorial of a number

**Faith**

it will give me fac(n-1);

4 X 3 X 2 X 1

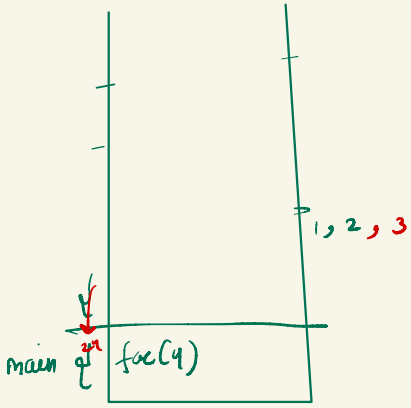
```
public static int fac(int n){
1  if(n==1) return 1
2  int sAns = fac(n-1);
3  int ans= n * sAns;
4  return ans;
}
```

**Expectation**

My function will return factorial of n

n = 5

5 X 4 X 3 X 2 X 1




ans = 2 X 1 ⇒ 2

ans = 3 X 2 = 6

ans = 4 X 6 ⇒ 24

---

**Ques** Calculate x raised to power n. **Recursively**

**Faith**

My function will calculate $x^{n-1}$ perfectly

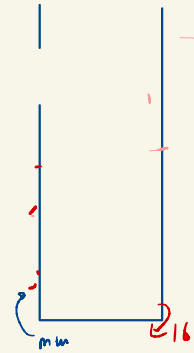
$2^4 \Rightarrow 2 \times 2 \times 2 \times 2$

```
public static int power(int x, int n){
    if(n==0) return 1;
1)  int sAns = power(x,n-1);
2)  int ans = x * sAns;
3)  return ans;
}
```

**Expectation**

My function will calculate x raised to power $x^n$

2 X 2 X 2 X 2 X 2




ans = x * 1 = 2 * 1

ans = 2 * 2 = 4

ans = 2 * 4 = 8

ans = 2 * 8 = 16

```java
public static int power(int x, int n){
1) if(n==0){
       return 1;
   }

2) int sAns = power(x,n-1);
3) int ans = x * sAns;

4) return ans;
}
```

psvi power (int x, int n) {

int sAns = power(x, n/2);
if (n%2 == 0)
ans = sAns * sAns
else
ans = x * sAns * sAns
}

Faith — My function will calculate $x^{\frac{n}{2}}$ perfectly.

$3^{\frac{6}{2}} \Rightarrow 3^3$
sAns = 3×3×3
ans = sAns * sAns

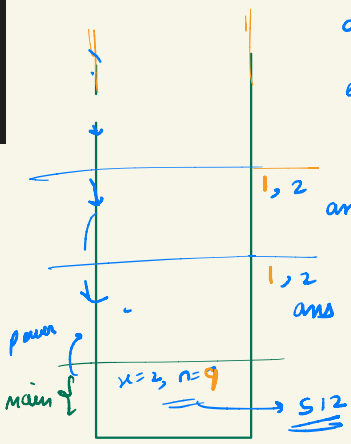
$x^{\frac{n}{2}} \Rightarrow 3^{\frac{1}{2}} \Rightarrow 3^3$
sAns = 3×3×3
ans = sAns × sAns × x

Expectation — My function will calculate $x^n$.

$3^6$ ans ⇒ (3×3×3) × (3×3×3)  (sAns, sAns)

$3^1$ ans ⇒ (3×3×3) × (3×3×3) × 3  (sAns, sAns, x)

```java
public static int power_log(int x, int n){
    if(n==0) return 1;
1) int sAns = power_log(x, n/2);
                          3, 5
2) int ans=0;
   if(n%2==0){ // if n is even
       ans = sAns * sAns;
   } else { // if n is odd
       ans = sAns * sAns * x;
   }

3) return ans;
}
```




ans = 1 * 1 × 2 ⇒ 2
ans = 2 * 2 ⇒ 4
$a^0 \Rightarrow 1$
ans = 4 * 4
ans = 16 * 16 * 2 = 512

on schedule
1 class

10:10 break

recursion in string

$\frac{n}{2^k} = 1$ → solve this for k.

Ques Sum of first n natural numbers recursively.

Faith

(n-1)

1+2+3+4

```java
// sum of first n natural numbers
public static int sum_natural(int n){
    if(n==0){
        return 0;
    }

    int sAns = sum_natural(n-1);
    int ans = n + sAns;

    return ans;
}
```

Expectation

n = 5

1+2+3+4+5

---

# Recursion in Arrays

Ques Print the array recursively.

Faith

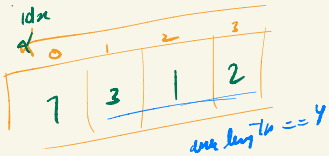
My fun. will print array from idx+1 to last.

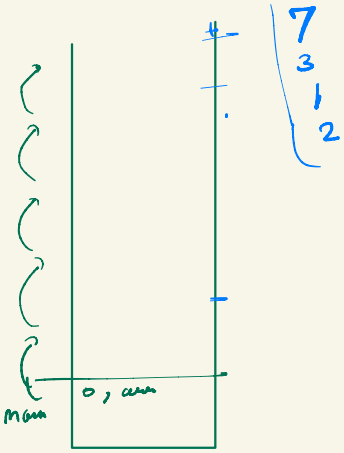
psv print(int idx, int[] arr){
Sysout(arr[idx]);
print(idx+1, arr);

Expectation

My function will print array from idx = 0 to end

3




3
1
2

Recursion on the way up

```java
public static void printArray(int idx, int[] arr){
    if(idx == arr.length){
        return;
    }
    System.out.println(arr[idx]);
    printArray(idx+1, arr);
}
```

→ why?

quality >>> quantity

Java to point

4 hours

15 min

b a b a b

if ( s > e)
return 1;

if ( char (s) ! ch(e)
return 0;

---

```
final class Complex {

private final double re;
private final double im;

public Complex(double re, double im) {
this.re = re;
this.im = im;
}

public String toString() {
return "(" + re + " + " + im + "i)";
}
}

class Main {
public static void main(String args[])
{
Complex c = new Complex(10, 15);
System.out.println("Complex number is " + c);
}
}
```

final ⇒ initialized only one time.

you can not change it once initialized

toString();

( 10 + 15i)

c → (10+15i)

Student s1 = new Student()

Student s1 {
String name = "Accio"

Public String toString(){
return name;
}

}

s1

Accio

class {

}

psv main () {

}

1 sec → $10^9$

Read operations

Read inputs

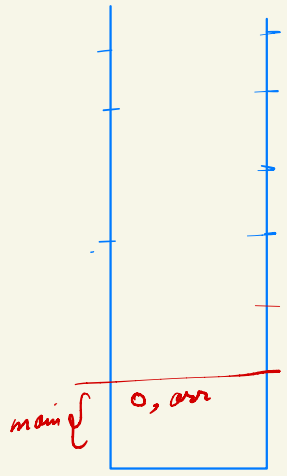
Read examples

Read

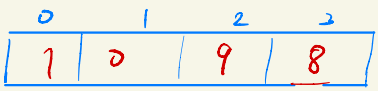
# Recursion with Arrays

recursion on the way up

```
// print array recursively ======================================
public static void printArray(int idx, int[] arr){
    if(idx == arr.length){
        return;
    }

    System.out.println(arr[idx]);
    printArray(idx+1, arr);
}
```




idx = 0




Qus Print array in reverse, idx = 0 from main.

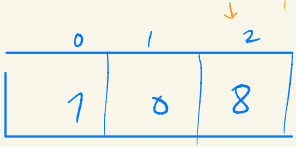
**Faith**

This function will print array in reverse from idx = 1 to end

```
psv reverseArray(int idx, int[] arr){
    reverseArray(idx+1, arr);
    System.out.println(arr[idx]);
}
```

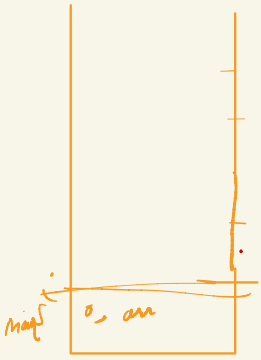
**Expectation**

This function will print array in reverse from idx = 0 to end




recursion on the way down

```
public static void printArrayReverse(int idx, int[] arr){
    if(idx==arr.length){
        return;
    }

    printArrayReverse(idx+1, arr);
    System.out.println(arr[idx]);
}
```



**Ques** Find maximum of an array recursively.

**Faith**

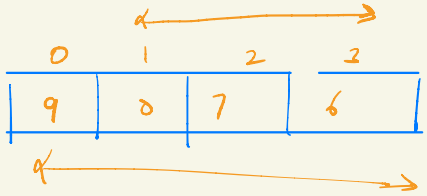
My function will return the max of array from idx=1 to end.

psvm

```
max(int idx, int[] arr){
    int sAns = max(idx+1, arr);
    int ans = Math.max(sAns, arr[idx]);
    return ans;
}
```

**Expectation**

My fun. will return the max of array from idx=0 to end.



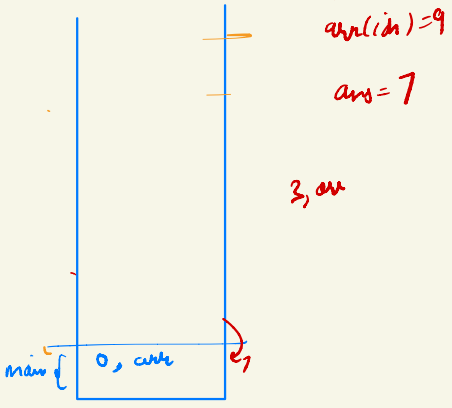

9

7

1) expectation
2) Faith → expectation but for a smaller value → call according to your faith.
3) work to reach your expectation from faith
4) dry run to find base case.

```
public static int maxOfArray(int idx, int[] arr){

1) int sAns = maxOfArray(idx+1, arr);
2) int ans=Math.max(arr[idx],sAns);

3) return ans;
};
```

```
if (idx == arr.length)
    return 0;
```




---

**Ques** Find the first index of target element

**Faith**

My function will return first index from idx=1 to end.

psvm

```
fi(int idx, int[] arr, int tar){
    int sAns = fi(idx+1, arr, tar);
    if (arr[idx] == tar){
        return idx;
    } else {
        return sAns;
    }
}
```

**Expectation**

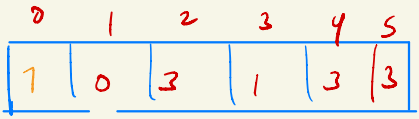
My function will returns first index of target from idx=0 to end

tar=3

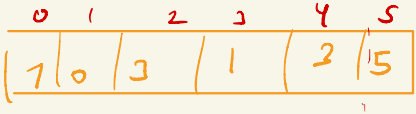
2

tar = 9

```
public static int firstIndex(int idx, int[] arr, int tar){

   1) int sAns = firstIndex(idx+1, arr, tar);
   2) int ans = 0;
      if(arr[idx] == tar){
          ans = idx;
      } else {
          ans = sAns;
      }

   3) return ans;
}
```




Ques Find the last index of given tar



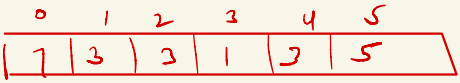

sAns = 1

```
public static int lastIndex(int idx, int[] arr, int tar){
   1) if(idx == arr.length){
          return -1;
      }

   2) int sAns = lastIndex(idx+1, arr, tar);
      int ans=0;

   3) if(sAns == -1){
          if(arr[idx]==tar){
              ans = idx;
          } else {
              ans = s[illegible]; -1;
          }
      } else {
          ans = sAns;
      }

      return ans;
}
```

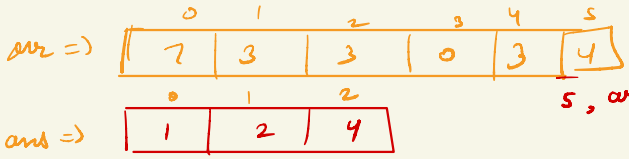
sAns = 4
ans = 4

Ques tar=3

ans = { 1, 2, 4 }



Ques Return an array containing indices of the target element.




```java
public static int[] allIndices(int idx, int[] arr, int tar, int count){
    if(idx == arr.length){
        int[] ba = new int[count];
        return ba;
    }

    int[] ans;
    if(arr[idx]==tar){
        ans = allIndices(idx+1, arr, tar, count+1);
    } else {
        ans = allIndices(idx+1, arr, tar, count);
    }

    if(arr[idx] == tar){
        ans[count] = idx;
    }

    return ans;
}
```

6, arr, 3, 3 → {0,0,0}
5, arr, 3, 3 → {0,0,0}
4, arr, 3, 2 → {0,0,4}
3, arr, 3, 2 → {0,0,4}
2, arr, 3, 1 → {0,2,4}
1, arr, 3, 0 → {1,2,4}
0, arr, 3, 0 → {1,2,4}

---

# Arraylist

```
Arraylist < String > al = new Arraylist <>();
```
(String: data-type; al: name)

⇒ add
```
al.add ("Akio");
```

⇒ get
```
al.get (idx);
```

⇒ size
```
al.size();
```

⇒ remove
```
al.remove (idx);
```

```
al.add ("a");
al.add ("b");
al.add ("c")
```

# Recursion in Strings

"Accio"

⇒ Substring
↳ part of a string (regular)

⇒ Subsequence ↳ delete any number of characters, you'd get a subsequence.

$\frac{n(n+1)}{2}$ abcd

$4 = 16$ $(2^n)$

substring

| | | | |
|---|---|---|---|
| a | b | c | d |
| ab | bc | cd | |
| abc | bcd | | |
| abcd | | | |

subsequence

| | | | |
|---|---|---|---|
| "" | b | c | d |
| a | bc | cd | |
| ab | bd | | |
| ac | bcd | | |
| ad | | | |
| abc | | | |
| abd | | | |
| acd | | | |
| abcd | | | |

---

abc ⇒ 8

"" ✓ b ✓ c ✓

a ✓ bc ✓

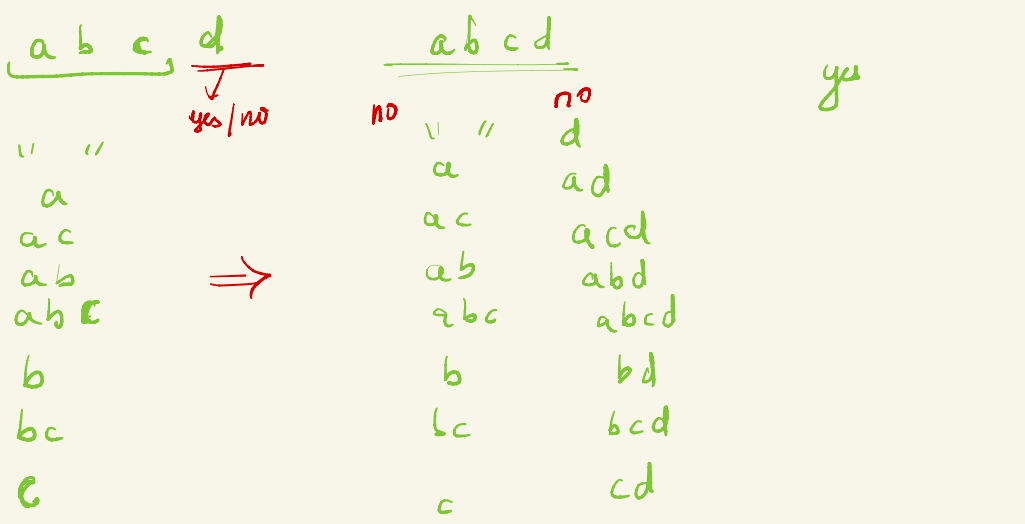
ac ✓

ab ✓

abc ✓

$2^n$
n = length of string

| a | b | c |
|---|---|---|
| yes/no | yes/no | yes/no |




**Ques** Return an arraylist containing all the subsequences of given string.

```
public static ArrayList<String> getSubsequences(String str){

}
```

str = abc

**Faith**

My function will return all the subs for "bc"

" "
b
c
bc

⇒

bc
" "
b
c
bc

abc
" " a
b ab
c ac
bc abc

SAns = { " ", b, c, bc }

**expectation**

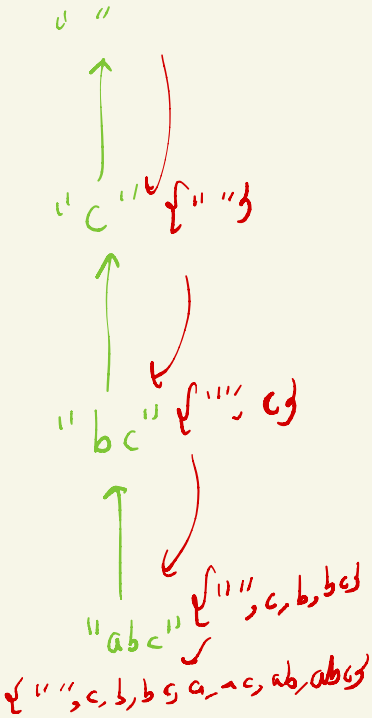
My function will return all the subs for "abc".

" " c
a ac
ab abc
b bc

ans = { " ", b, c, bc, a, ab, ac, abc }

```java
public static ArrayList<String> getSubsequences(String str){

    char first_char = str.charAt(index: 0);
1)  String ss = str.substring(beginIndex: 1); // small string

2)  ArrayList<String> sAns = getSubsequences(ss); // faith

    ArrayList<String> ans = new ArrayList<>();

    // copy from sAns
3)  for(int i=0; i<sAns.size(); i++){
        String sub = sAns.get(i);
        ans.add(sub);
    }

    // add first char to every sAns element
4)  for(int i=0; i<sAns.size(); i++){
        String sub = sAns.get(i);
        String ansSub = first_char + sub;
        ans.add(ansSub);
    }

    return ans;
}
```

break till 10:05




0 1 1 2 3 5 8 13 21 34

1 1 2 3 5 8 13 21 34 55 89 144

$f(n) = f(n-1) + f(n-2)$

$\Rightarrow f(n-2) + f(n-3) + f(n-3) + f(n-4)$

$\Rightarrow f(n-2) + 2f(n-3) + f(n-4)$;

$\Rightarrow f(n-3) + f(n-4) + 2f(n-3) + f(n-4)$;

$\Rightarrow 3f(n-3) + f(n-4) + f(n-4)$

$\Rightarrow 3f(n-3) + f(n-5) + f(n-6) + f(n-4)$

$\Rightarrow 4f(n-3) + f(n-6)$

$F(E) \Rightarrow 4 * F_{E-1} + F_{E-2}$

$F_E = 4f(E-1) + f(E-2)$

$fE_0 = 0$

$fE_1 = 2$

| 1 | 3 | 5 | 1 | 4 | 2 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

```java
public static int robHouse(int idx, int end, int[] arr){
  if(idx>end){
    return 0;
  }

  // rob current idx
    int rob = robHouse(idx+2,end,arr) + arr[idx];

  // skip current idx
  int skip = robHouse(idx+1, end, arr);

  int ans = Math.max(rob,skip);
  return ans;
}
```