Functions ⇒ Takes input and give some output.

$f(a, b) = a^2 + 2ab + b^2$

$f(x) = 2x + 1$

$f(n) = 2n + 1$

$x = 5$ → input

output $2 * 5 + 1 \Rightarrow 11$

input = 5

output = 11

$(n = 5)$

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

public static void fsum ( int a) {
```

return type → int

parameters / arguments

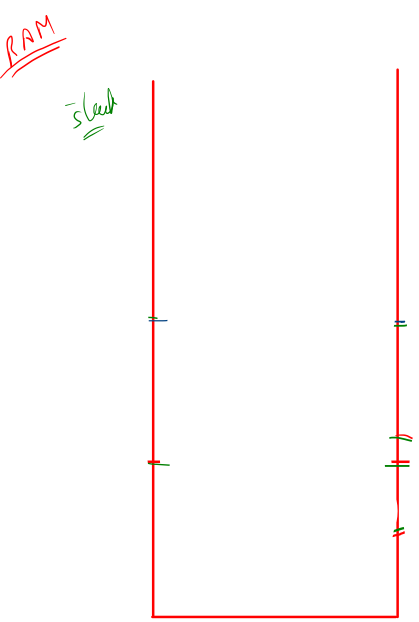
no return type → void

sum (2, 3) = 5

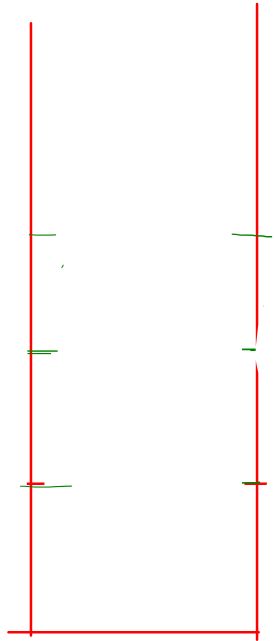
sum (1, 4) = 5

sum (9, 11) = 20

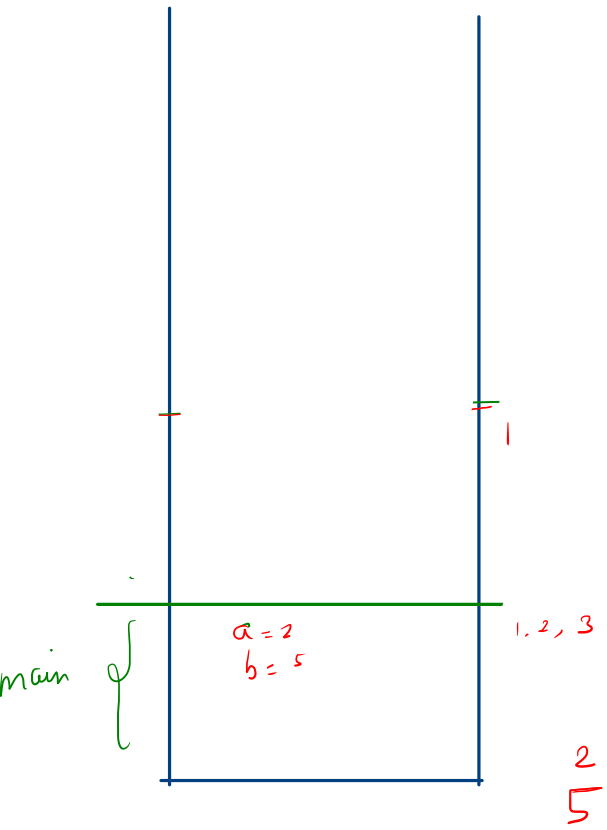
}




```java
class Main {
    public static int sum(int a, int b){
        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);
    }
}
```



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

    public static void fun(int a, int b){
        int sum = a+b;
        System.out.println(sum);

        System.out.println("I am inside a void function");
        System.out.println("I am inside a void function");
    }

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

        fun(a,b);
    }
}
```




```java
class Main {
    public static void swap(int a, int b){
        int temp = a;
        a = b;
        b = temp;
    }
    Run | Debug
    public static void main(String[] args) {
        int a= 2;
        int b= 5;

        swap(a, b);

        System.out.println(a);
        System.out.println(b);
    }
}
```

Recursion

→ How to solve any recursion problem

1) Keep a faith that it works for a smaller problem. (faith)
2) Solve for the smallest problem.
3) Solve for the actual problem (to meet your expectation)

Ques Print decreasing numbers from 'n' to 1 using recursion.

n = 5

Expectation 5 4 3 2 1

```
public static void printDecreasing(int n){
    System.out.println(n);
    printDecreasing(n-1);
}
```

Faith (n-1)

function will print numbers from (n-1) to 1

n-5 → 4 3 2 1

Expectation (n)

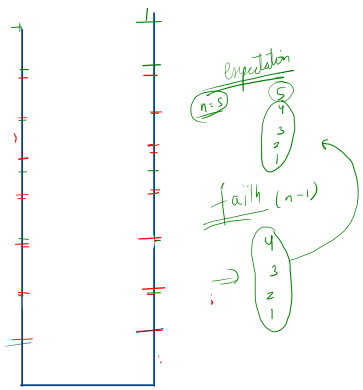
function will print numbers from 'n' to 1

n=5 → 5 4 3 2 1

work

you won't be able to learn recursion without dry-run-

stack overflow




```
class Main {
    public static void printDecreasing(int n){
        if(n==0){
            return;
        }
        System.out.println(n);
        printDecreasing(n-1);
    }
    Run | Debug
    public static void main(String[] args) {
        int n = 3;
        printDecreasing(n);
    }
}
```

3
2
1

Ques Print numbers in increasing order using recursion.

Faith (n-1)

It will work for (n-1)
print numbers from 1 to 'n-1'

1
2
3
4

5

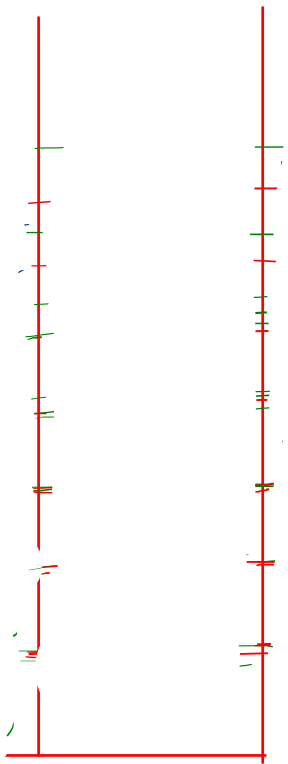
```
psv printIncreasing(int n){

    printIncreasing(n-1);
    System.out.println(n);

}
```

Expectation (n)

Function will print numbers from 1 to n

n=5

1
2
3
4
5



```java
public static void printIncreasing(int n){
    if(n==0){
        return;
    }
    printIncreasing(n-1);
    System.out.println(n);
}
Run | Debug
public static void main(String[] args) {
    int n = 3;
    printIncreasing(n);
}
```

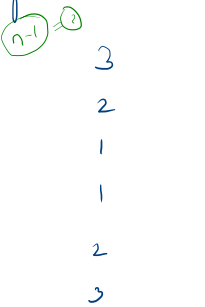
1
2
3

Ques Print numbers in decreasing order, then increasing using recursion.

n=5

Faith (n-1)

It will print numbers from (n-1) to 1 and then 1 to (n-1)



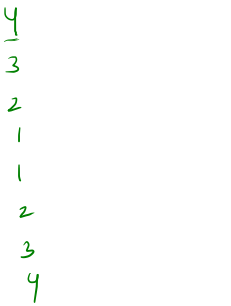

```
psv pdi(int n){
    System.out.println(n);
    pdi(n-1);
    System.out.println(n);
}
```

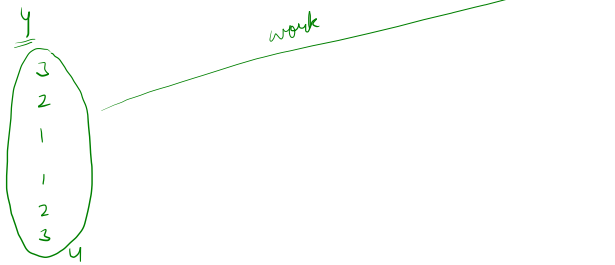
fail

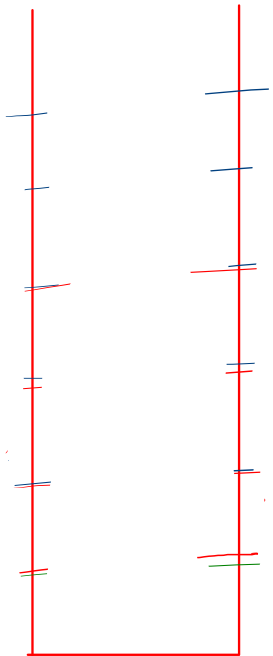
Expectation

It will print numbers in decreasing then increasing order

n=4







```java
public static void pdi(int n){
    if(n==0){
        return;
    }
    System.out.println(n);
    pdi(n-1);
    System.out.println(n);
}
Run | Debug
public static void main(String[] args) {
    int n = 4;
    pdi(n);
}
```

Ques Find factorial of given number 'n'.

Faith (n-1)

It will return factorial (n-1)

1 x 2 x 3 x 4

smallAns

```
public static int fac(int n){
    int smallAns = fac(n-1);
    int ans = smallAns * n;
    return ans;
}
```

faith work

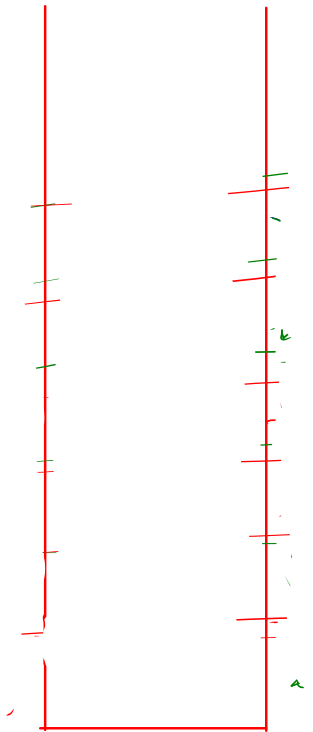
Expectation

It will return factorial of given number

n = 5

1 x 2 x 3 x 4 x 5

work



24

```
public static int factorial(int n){
    if(n==0){
        return 1;
    }
    int smallAns = factorial(n-1);
    int ans = smallAns * n;
    return ans;
}
Run | Debug
public static void main(String[] args) {
    int n = 4;
    int ans = factorial(n);
    System.out.println(ans);
}
```

Ques Find x^n using recursion

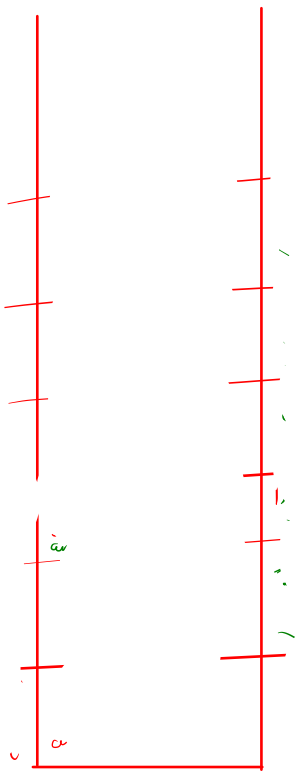
Faith (n-1)
It will give us p(x, n-1)

```
public static int power (int x, int n) {
    int smallAns = power (x, n-1);
    int ans = smallAns * x;
    return ans;
}
```

Expectation
It will return x^n.
x=2, n=4

2x2x2x2 (16)

2x2x2
8



x^n

if (n==0)
return 1

x^0

16

```
public static int power(int x, int n){
    if(n==0){
        return 1;
    }
    int smallAns = power(x, n-1);
    int ans = smallAns * x;
    return ans;
}
Run | Debug
public static void main(String[] args) {
    int x = 2;
    int n = 4;
    int ans = power(x,n);
    System.out.println(ans);
}
```