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

    int xrnb2 = power(x,n/2);
    int xrn = xrnb2 * xrnb2;

    if(n % 2 == 1) {
        //n is odd
        xrn = xrn * x;
    }

    return xrn;
}
```

$x = 2$

$n = 8$

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

    int xrn = power(x,n/2)* power(x,n/2);

    if(n % 2 == 1) {
        //n is odd
        xrn = xrn * x;
    }

    return xrn;
}
```

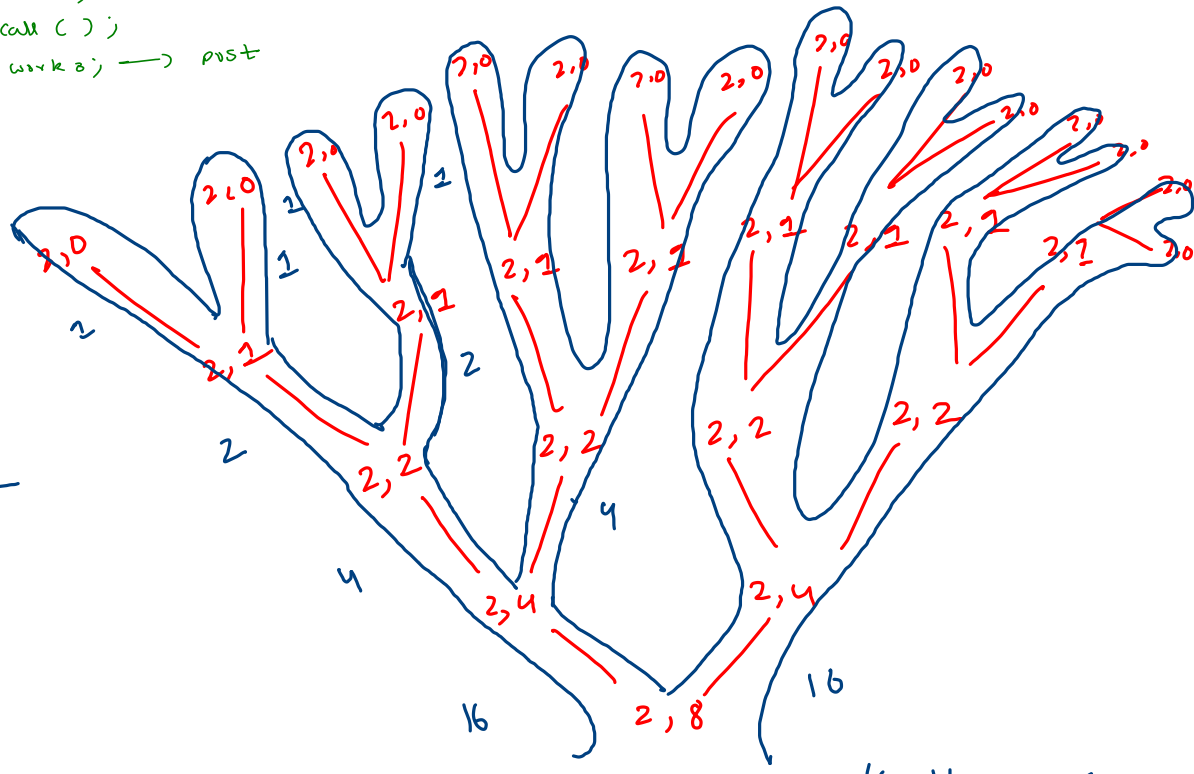
~~power~~	~~$x=2, n=0$~~	
~~power~~	~~$x=2, n=1, x^{\frac{n}{2}}=1, x^n=2$~~	1 2 3 4
~~power~~	~~$x=2, n=2, x^{\frac{n}{2}}=2, x^n=4$~~	1 2 4
~~power~~	~~$x=2, n=4, x^{\frac{n}{2}}=4, x^n=16$~~	1 2 4
~~power~~	~~$x=2, n=8, x^{\frac{n}{2}}=16, x^n=256$~~	1 2 4

~~power~~	~~$x=2, n=0$~~	
~~power~~	~~$x=2, n=0$~~	
~~power~~	~~$x=2, n=1, x^n=1*1*2$~~	1 2 3
power	$x=2, n=2, x^n=2*$	1
power	$x=2, n=4, x^n$	1
power	$x=2, n=8, x^n=$	1

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

    int xrn = power(x,n/2)* power(x,n/2);

    if(n % 2 == 1) {
        //n is odd
        xrn = xrn * x;
    }

    return xrn;
}
```

power	x=2, n=2	
power	x=2, n=0	
power	x=2, n=0	
power	x=2, n=1, 1*1×2	1
power	x=2, n=2, 2*	1
power	x=2, n=4	1
power	x=2, n=8,	1

work2; → pre
call ( );
work2; → in
call ( );
work3; → post



```
Input2 -> 3
Output3 -> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3
```

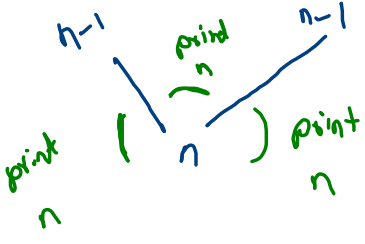
```
Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2
```

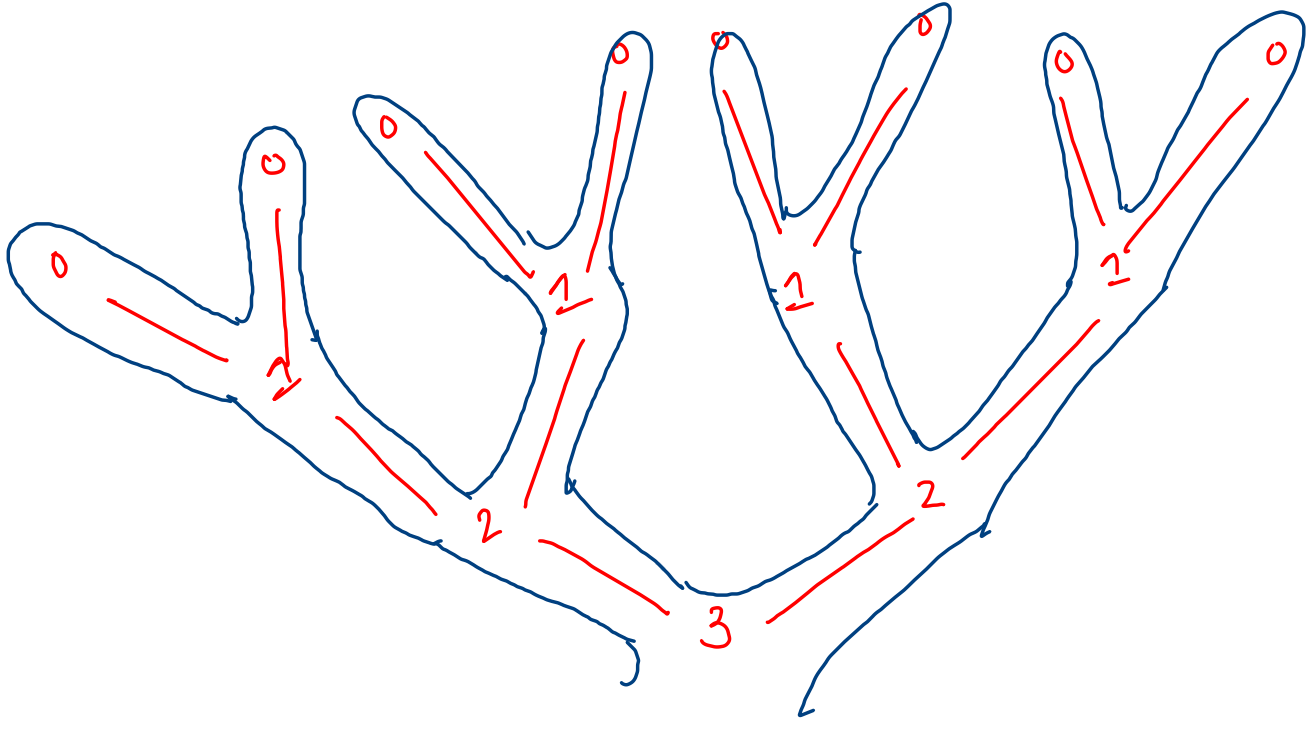
expectation pzz(3) = 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

faith pzz(2) = 2 1 1 1 2 1 1 1 2

pzz(3) = 3 + pzz(2) + 3 + pzz(2) + 3

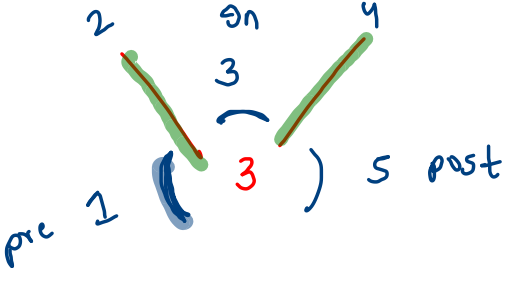
```java
public static void pzz(int n){
    if(n == 0) {
        return;
    }

    System.out.print(n + " ");
    pzz(n-1);
    System.out.print(n + " ");
    pzz(n-1);
    System.out.print(n + " ");
}
```





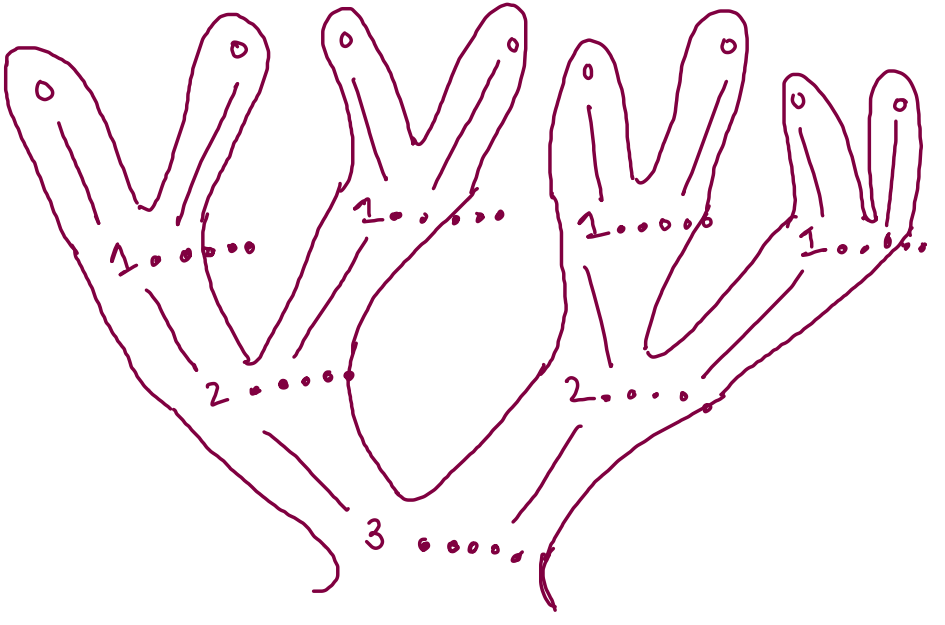
3	2
2	1
1	1
1	1
1	1
2	2
1	1
1	1
1	1
2	2
3	3

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

    System.out.print(n + " ");
    pzz(n-1);
    System.out.print(n + " ");
    pzz(n-1);
    System.out.print(n + " ");
}
```



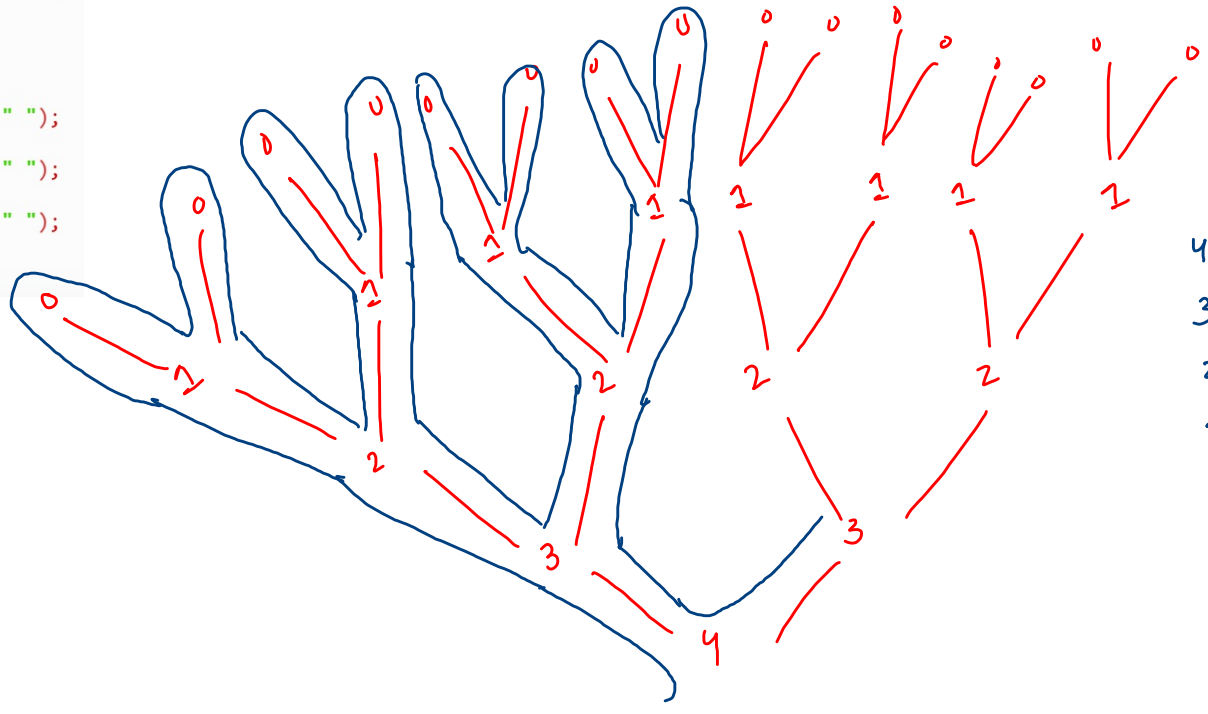
line no.

n = 4



3	2
2	1
1	1
1	1
1	2
2	1
1	1
1	1
1	2
2	3
3	

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

    System.out.print(n + " ");
    pzz(n-1);
    System.out.print(n + " ");
    pzz(n-1);
    System.out.print(n + " ");
}
```



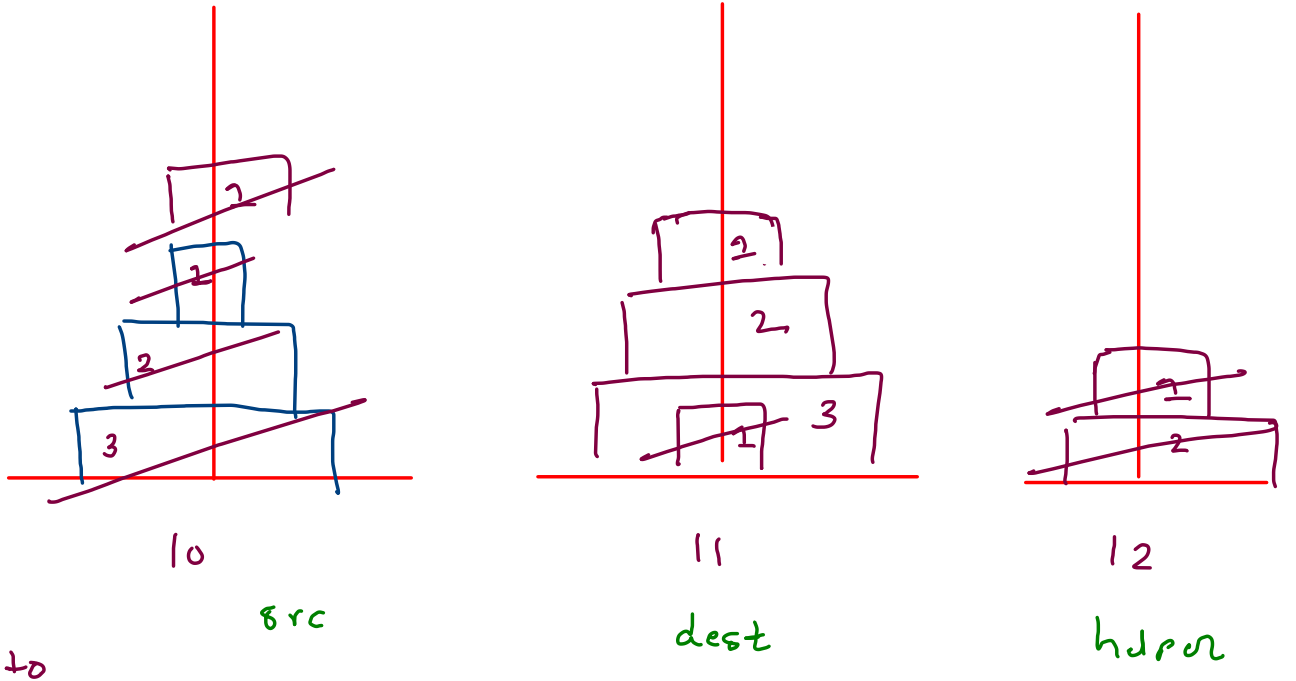
4	2	2
3	3	3
2	2	4
1	1	
1	1	
1	1	
2	2	
1	1	
1	1	
1	1	

1 + 22

= 45

$n = 3$

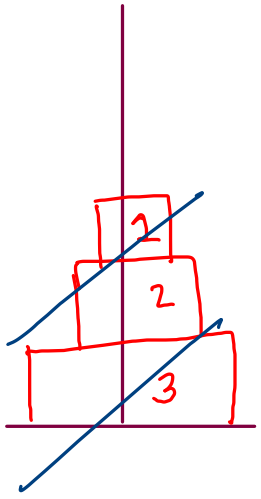
- ✓ 1[10 -> 11]
- ✓ 2[10 -> 12]
- ✓ 1[11 -> 12]
- ✓ 3[10 -> 11]
- ✓ 1[12 -> 10]
- ✓ 2[12 -> 11]
- ✓ 1[10 -> 11]



(i) move one disk at a time.

(ii) it is not allowed to place heavier disk on lighter disk.

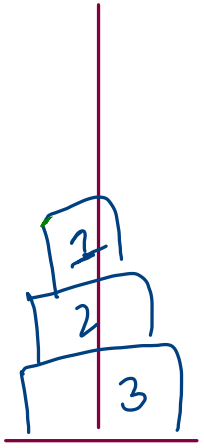
faith: move some disks from "any source to any dest"



10

(src)

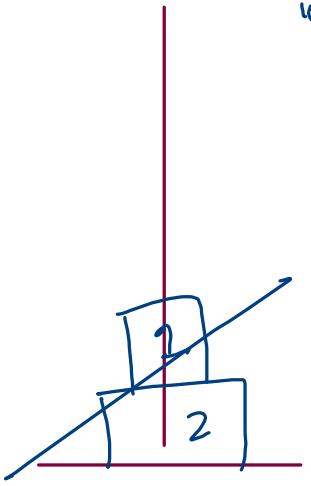
t1 gd



11

(dest)

t2 gd



12

(helper)

t3 gd

(i) move n-1 disk from t1 to t3 faith1 (src to helper).

(ii) move n^{th} disk from t1 to t2. (your step)

(iii) move n-1 disk from t3 to t2 faith2

```java
public static void toh(int n, int t1id, int t2id, int t3id){
    if(n == 0) {
        return;
    }

    //to move n-1 disks from t1 to t3 using t2
    toh(n-1,t1id,t3id,t2id);

    //my work - to move nth disk from t1 to t2
    System.out.println(n + "[" + t1id + " -> " + t2id + "]");

    //to move n-1 disks from t3 to t2
    toh(n-1,t3id,t2id,t1id);

}
```

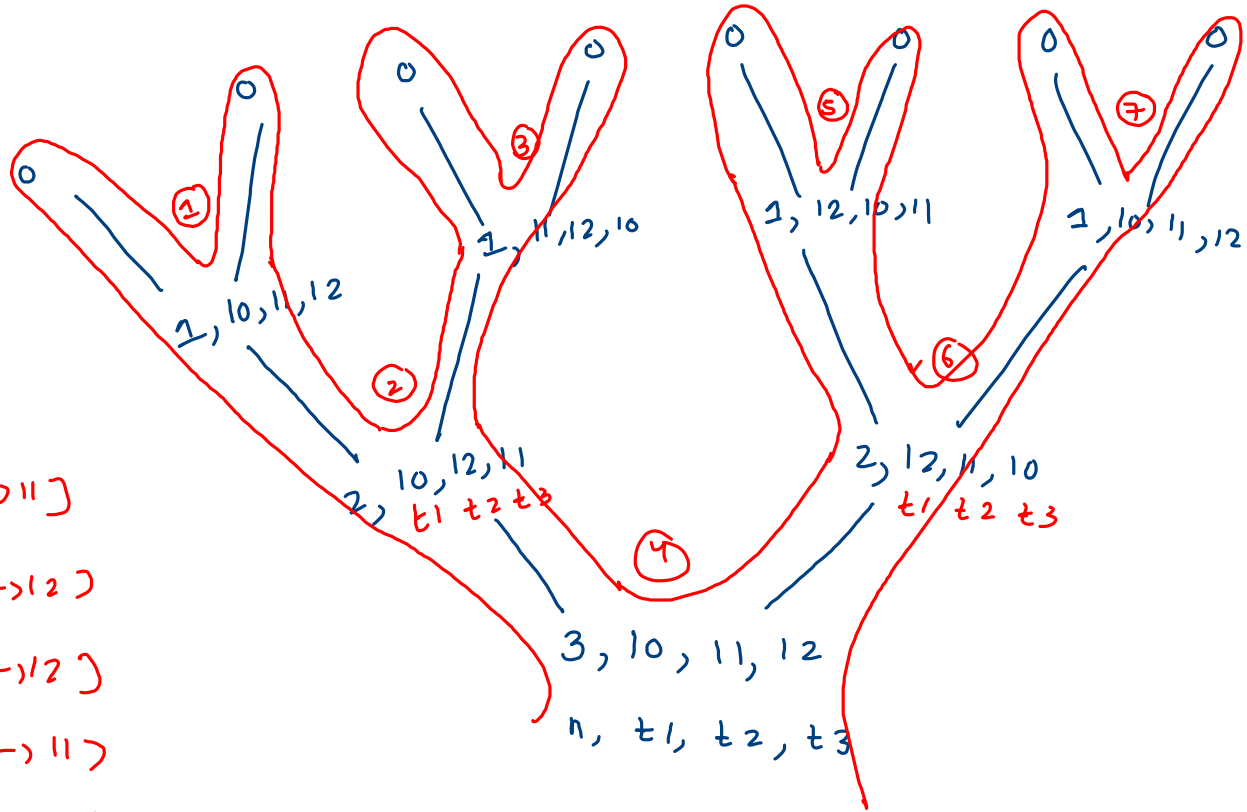
n-1, t1, t3, t2

work

n, t1, t2, t3

n-1, t3, t2, t1

n = 4



1 [10 -> 11]

2 [10 -> 12]

1 [11 -> 12]

3 [10 -> 11]

1 [12 -> 10]

2 [12 -> 11]

1 [10 -> 11]

i=0

10	20	30	40	50
0	1	2	3	4

4k

i=1

~~10~~	20	30	40	50
0	1	2	3	4

4k

i=2

~~10~~	~~20~~	30	40	50
0	1	2	3	4

4k

i=3

~~10~~	~~20~~	~~30~~	40	50
0	1	2	3	4

	10	20	30	40	50
4K	0	1	2	3	4

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

1   System.out.println(arr[idx]);
2   displayArr(arr,idx+1); //faith -> to print array from idx+1 till end
}
```

10 20 30 40 50

dA	arr = 4K, idx = 5	
dA	arr = 4K, idx = 4	1 2
dA	arr = 4K, idx = 3	1 2
dA	arr = 4K, idx = 2	1 2
dA	arr = 4K, idx = 1	1 2
dA	arr = 4K, idx = 0	1 2

10	20	30	40	50
0	1	2	3	4

arr

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

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

1

2

50 40 30 20 10

dra	arr = arr, idx = 5	
dra	arr = arr, idx = 4	1 2
dra	arr = arr, idx = 3	1 2
dra	arr = arr, idx = 2	1 2
dra	arr = arr, idx = 1	1 2
dra	arr = arr, idx = 0	1 2

17	20	12	24	18
0	1	2	3	4

uic

faith

(smax)

→ omax → math.max

(smax, arr[idx])

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

1   int smax = maxOfArray(arr,idx+1);  //smaller array max

2   int omax = Math.max(smax,arr[idx]);

3   return omax;
}
```

~~max~~	~~idx = 4~~	
~~max~~	~~idx = 3~~, sm = 18, om = 24	1 2 3
~~max~~	~~idx = 2~~, sm = 24, ~~om = 24~~	~~1 2 3~~
~~max~~	~~idx = 1~~, ~~sm = 24~~, om = 24	1 2 3
~~max~~	~~idx = 0~~, ~~sm = 24~~, om = 24	1 2 3

24

$x = 13$

29	40	12	24	13	25	13	28
0	1	2	3	4	5	6	7

val = 15

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

    if(arr[idx] == x) {
        return idx;
    }
    else {
        int fis = firstIndex(arr,idx+1,x);
        return fis;
    }
}
```

1

2

FI	idx = 4, x = 13	1
FI	idx = 3, x = 13, fis = 4	2
FI	idx = 2, x = 13, fis = 4	2
FI	idx = 1, x = 13, fis = 4	2
FI	idx = 0, x = 13, fis = 4	2

4

last index

29	40	13	24	20	13	25	28
0	1	2	3	4	5	6	7

x = 13

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

1   int lis = lastIndex(arr,idx+1,x); //last index in smaller array

a   if(lis != -1) {
        return lis;
    }
b   else if(arr[idx] == x) {
        return idx;
    }
    else {
c       return -1;
    }
}
```

It	idx = 8	
It	idx = 7, lis = -1	1 c
It	idx = 6, lis = -1	1 c
It	idx = 5, lis = -1	1 b
It	idx = 4, lis = 5	1 a
It	idx = 3, lis = 5	1 a
It	idx = 2, lis = 5	1 a
It	idx = 1, lis = 5	1 a
It	idx = 0, lis = 5	1 a

5