Recursion

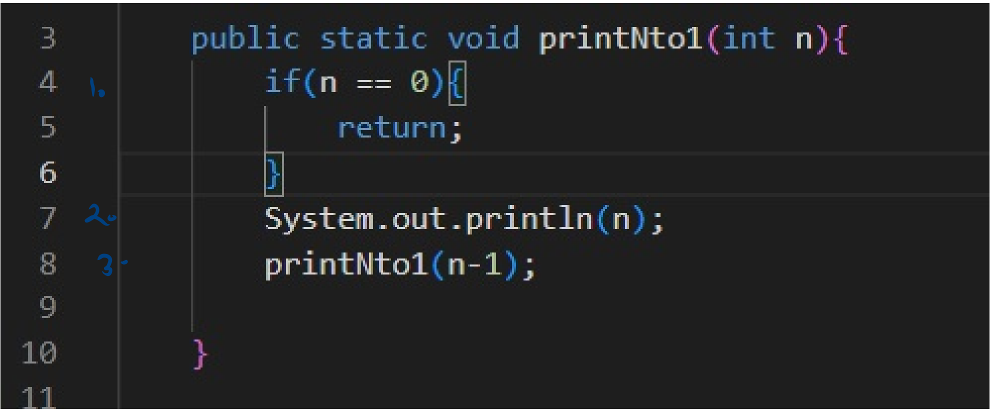
Q1.

→ n = 5

5 ✓
4
3
2
1

5
4
3
2
1

```
if ( n == 0)
    return;

print( n) ✓
fun( n-1)
```



```
    public static void printNto1(int n){
1.      if(n == 0){
            return;
        }
2.      System.out.println(n);
3.      printNto1(n-1);

    }
```

n=4

4
3
2
1

Q2

5

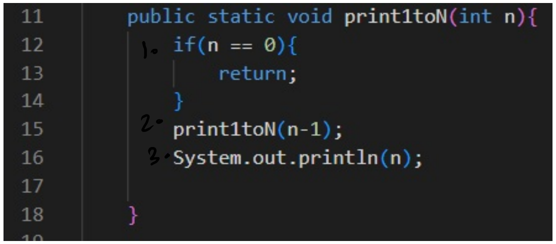
1

2

3

4

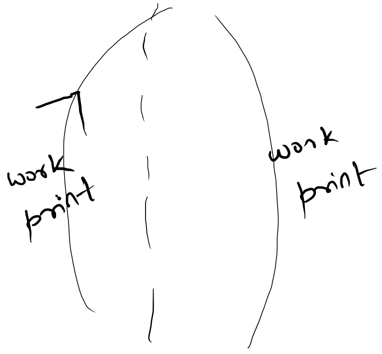
5

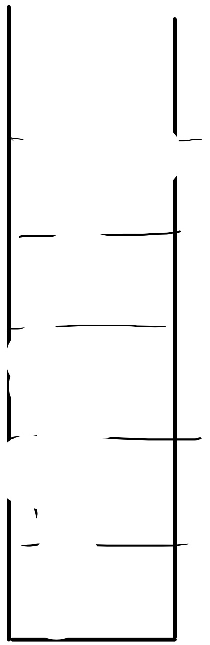


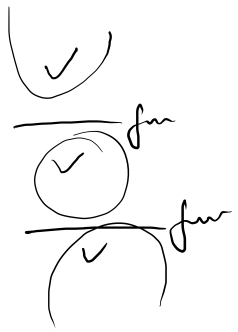
```
public static void print1toN(int n){
    1. if(n == 0){
            return;
        }
    2. print1toN(n-1);
    3. System.out.println(n);

}
```

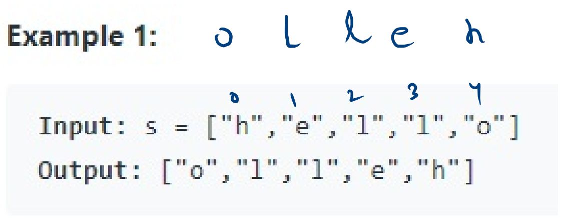
n=4

1
2
3
4









Example 1: o l l e h

```
         0    1    2    3    4
Input: s = ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]
```

$n = 5$

$\underline{\underline{i > n/2}} \rightarrow$ return.

$\boxed{n - 1 - i}$

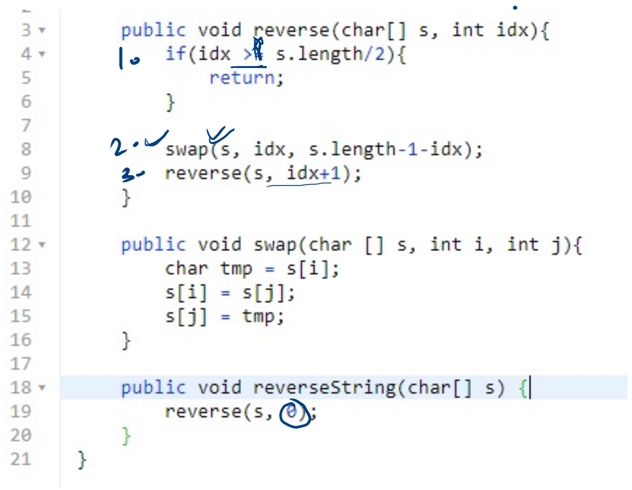
$i = 1$

$\uparrow$

$i = 0$

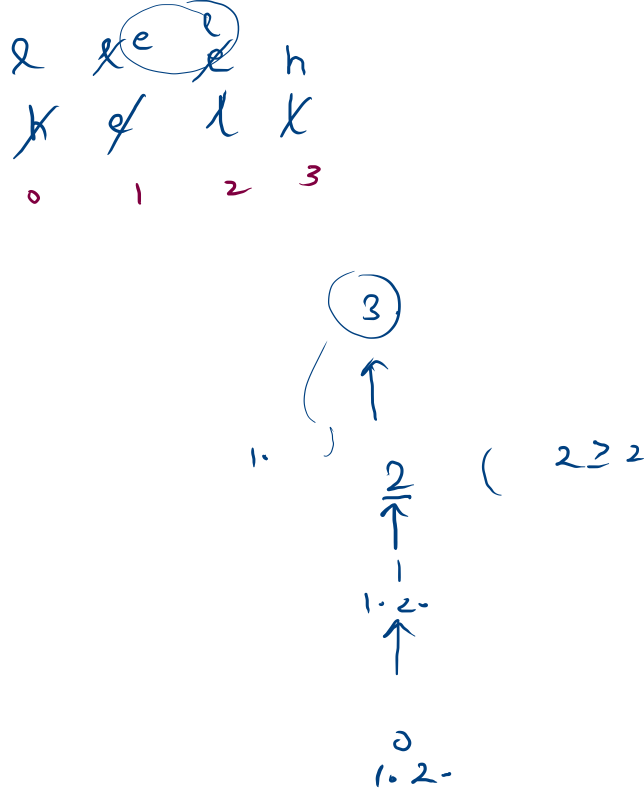
swap (0, 4)

n-1-i



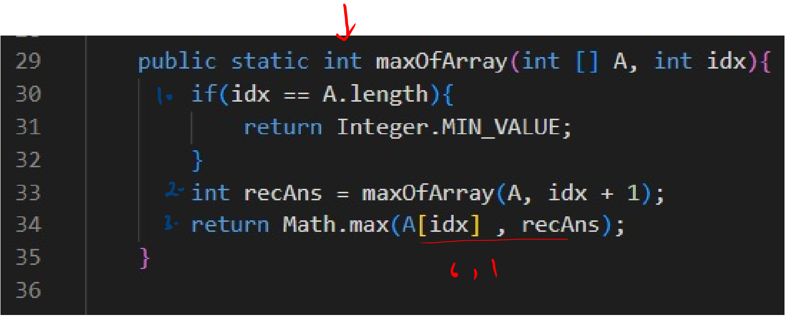
```
    public void reverse(char[] s, int idx){
1.      if(idx >= s.length/2){
            return;
        }

2.      swap(s, idx, s.length-1-idx);
3.      reverse(s, idx+1);
    }

    public void swap(char [] s, int i, int j){
        char tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }

    public void reverseString(char[] s) {
        reverse(s, 0);
    }
}
```

2>2

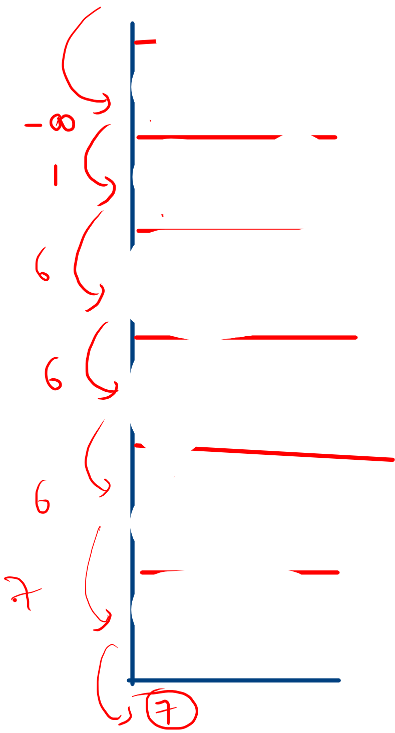


max
ans = 7

2	7	5	3	6	1
0	1	2	3	4	5



```java
    public static int maxOfArray(int [] A, int idx){
        if(idx == A.length){
            return Integer.MIN_VALUE;
        }
        int recAns = maxOfArray(A, idx + 1);
        return Math.max(A[idx] , recAns);
    }
```



Que.

2 3 2 2 2 1 4 2 5 2

key = 2.

freq of key.

Que.

$n = 5 \rightarrow 5! = 5 \times 4 \times 3 \times 2 \times 1$

$= 120$