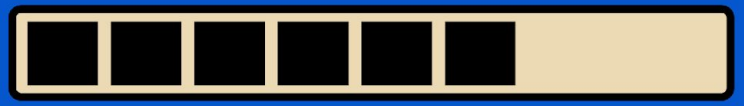
SOFTWARE INCUBATOR

PRESENTS

BitSet



Competitive programming Workshop...

Bit Magic

Bit Manipulation

Bit manipulation is the process of manipulating bit of a data.
Eg : 7 can be represented as 111 (in binary) & 5 can be represented as 101

$$1\ 1\ 1 = 1 * 2^2 + 1 * 2^1 + 1 * 2^0 = 7$$
$$1\ 0\ 1 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 5$$

AND

5 & 6 = 4

101 & 110 = 100

Tricks

- a & a = a
- a & 1 = a % 2
- a & 0 = 0

X	Y	AND
0	0	0
0	1	0
1	0	0
1	1	1

OR

5 | 6 = 7

101 | 110 = 111

Tricks

- b | b = b
- a | 0 = a

X	Y	OR
0	0	0
0	1	1
1	0	1
1	1	1

NOT

~ 5 = 2
~ 101 = 010

~ 2 = 5
~ 010 = 101

Trick

~(~a) = a

X	~X
0	1
1	0

XOR

5 ^ 6

101 ^ 110 = 011 => $(011)_2 = 3$

5 ^ 6 = 3

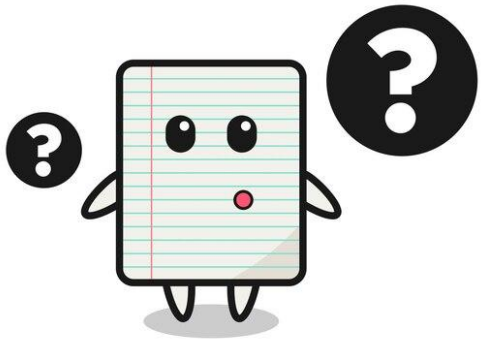
Tricks

- Exist in triplet
  - 5 ^ 6 = 3
  - 5 ^ 3 = 6
  - 3^ 6 = 5
- b ^ b = 0
- b ^ 0 = b

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Question

For each given number N, your task is to determine whether there are more ones (1s) than zeros (0s) in its binary form.



https://bit.ly/4jpDahm

Left Shift

7 << 2

$7 = (111)_2$

11100

$(11100)_2 = 28$

7 << 2 = 28

Tricks

Multiplication with powers

of 2

$a << 1 = 2 * a$

$a << 4 = 16 * a$

$a << k = 2^k * a$

Right Shift

7 >> 2

$7 = (111)_2$

$(00111)_2 = 7$

001

$(001)_2 = 1$

7 >> 2 = 1

Tricks

Integer division with powers

of 2

a >> 1 = a / 2

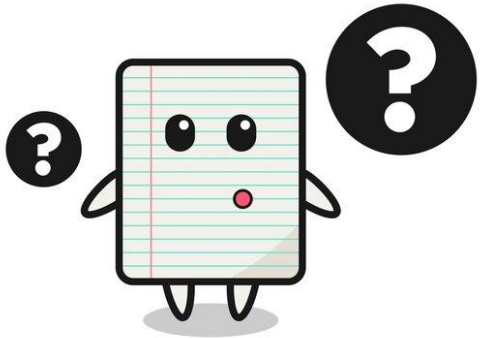
a >> 4 = a / 16

a >> k = a / 2^k

Question

You are given an integer N. Your task is to find two numbers A and B such that:

1. **0 ≤ A, B ≤ N**
2. **A ≠ B**
3. **A XOR B is maximized**



https://bit.ly/3GaH67p

RECURSION

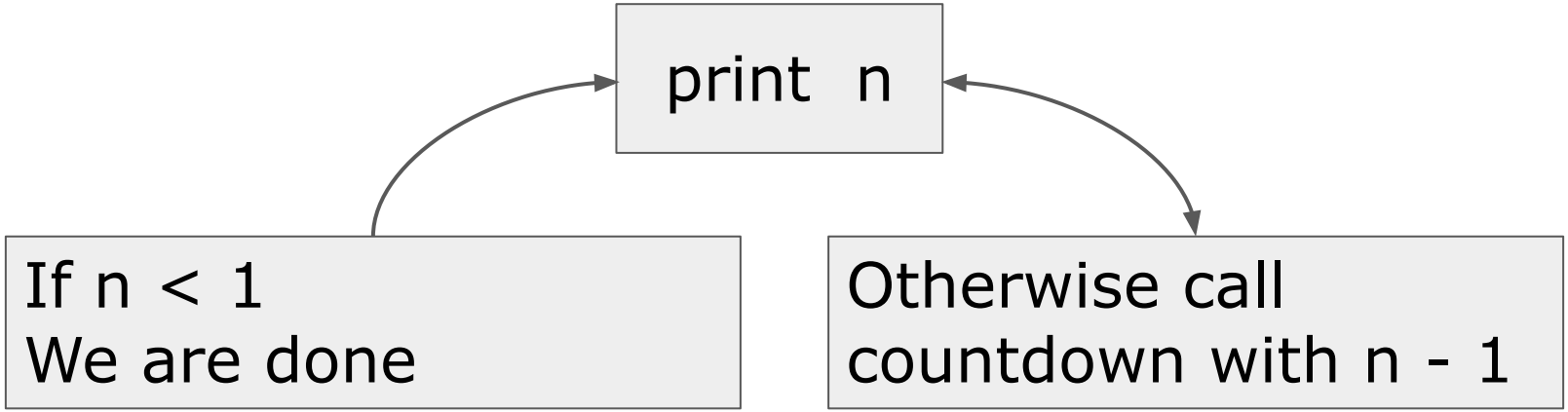
Recursion

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function.

Example:-

```
void function()
{
    function();
}
```

SOFTWARE INCUBATOR




```
int recursion(int n){
	if(n == 0) return 0;

	return n+recursion(n-1);
}
int main(){
	cout<<recursion(5);
	return 0;
}
```

```
void fun()
{
    // base condition
    if(condition)
        return;

    // recursive call
    fun();
}
```

Factorial

$5! = 5 * 4 * 3 * 2 * 1$

$4! = 4 * 3 * 2 * 1$

$5! = 5 * 4!$

$n! = n * (n-1)!$

Print factorial

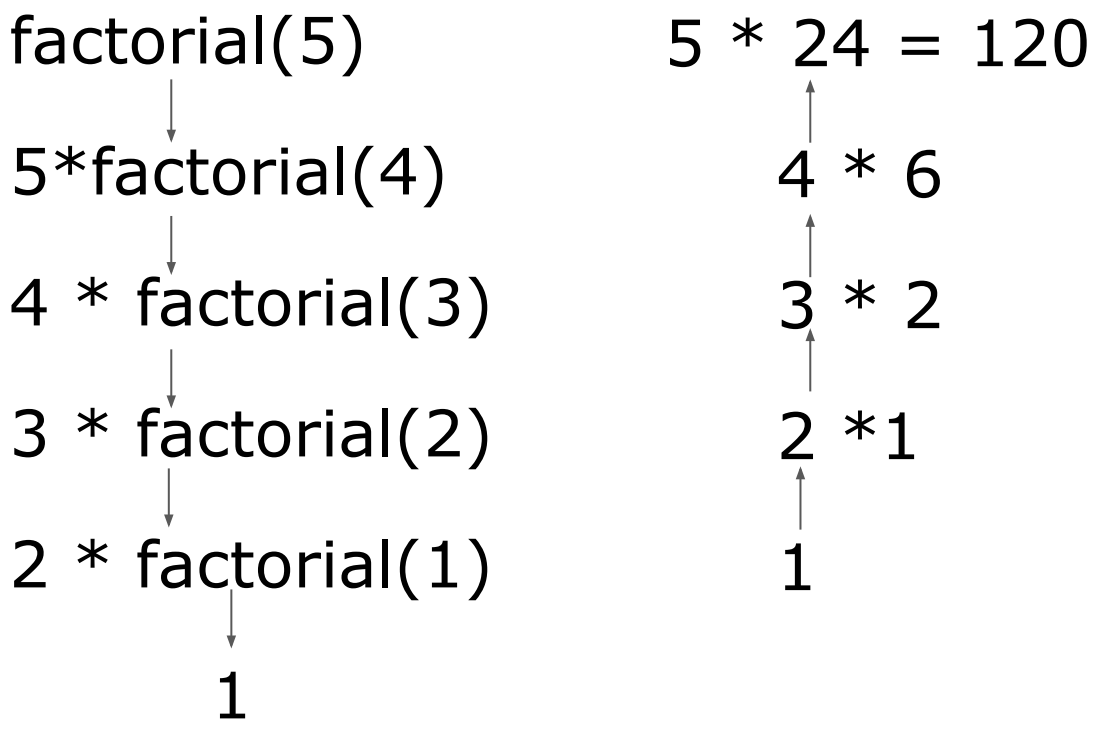
```
#include<bits/stdc++.h>
using namespace std;
int factorial(int n) {
        if (n==1 || n==0)                    // base conditi
                return 1;
        else
                return n*factorial(n-1);   // recursive case
}
int main() {
   int n;
   cin>>n;
   int result = factorial(n);
   cout<<result<<"\n";
   return 0;
}
```

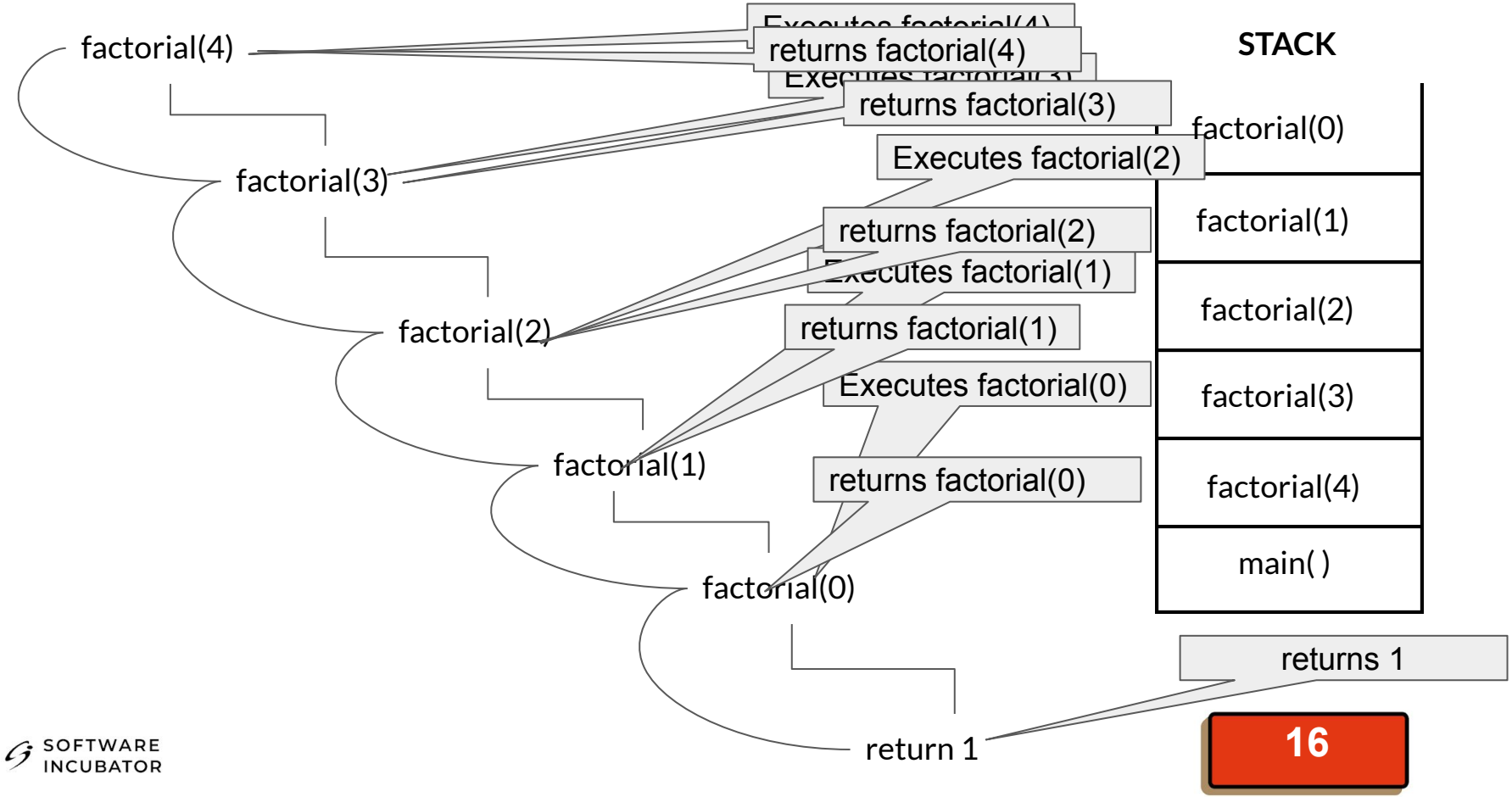
INPUT

5

OUTPUT

120


factorial(5)
5*factorial(4)
4 * factorial(3)
3 * factorial(2)
2 * factorial(1)
1
5 * 24 = 120
4 * 6
3 * 2
2 *1
1


factorial(4)
returns factorial(4)
returns factorial(3)
factorial(3)
Executes factorial(2)
returns factorial(2)
factorial(2)
returns factorial(1)
Executes factorial(0)
factorial(1)
returns factorial(0)
factorial(0)
returns 1
return 1
STACK
factorial(0)
factorial(1)
factorial(2)
factorial(3)
factorial(4)
main( )
16

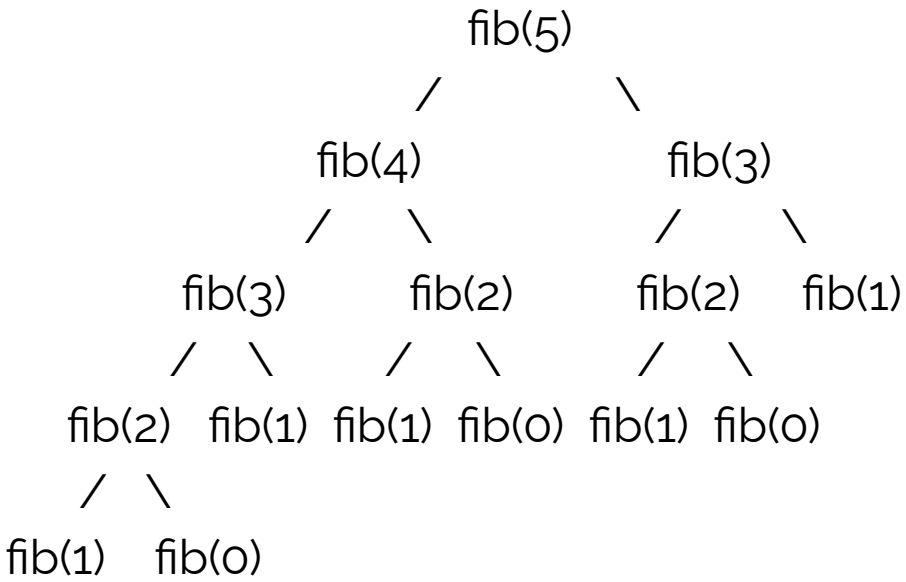
Fibonacci

1st = 0
2nd = 1
3rd = 2nd + 1st = 1 + 0 = 1

nth = (n-1)th + (n-2)th

=> 0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 . . .

Recursion Tree



Print the nth fibonacci

```
#include<bits/stdc++.h>
using namespace std;
int fib(int n) {
        if(n==1)       return 0;
        else if(n==2)       return 1;
        else    return fib(n-1)+fib(n-2);
}
 int main() {
    int n;
    cin >> n;
    int result = fib(n);
    cout<<result<<"\n";
    return 0;
 }
```

INPUT

5

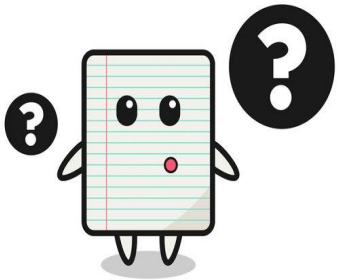
OUTPUT

3

SOFTWARE INCUBATOR

Question

You are given an integer N representing the number of steps in a staircase. You can climb either 1 step or 2 steps at a time. Your task is to determine the total number of distinct ways to reach the top of the staircase.

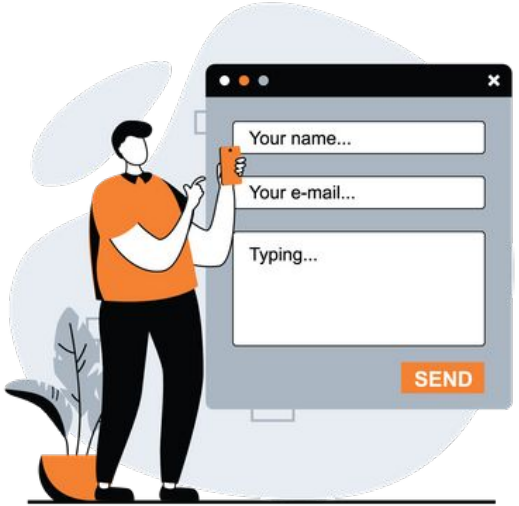


https://bit.ly/3G5IrfA

SOFTWARE INCUBATOR

Feedback Form

https://bit.ly/42zu7nm



Thank You