



**CHANDIGARH
UNIVERSITY**
Discover. Learn. Empower.

DOMAIN WINTER CAMP

(Department of Computer Science and Engineering)

Name- Aman Bansal UID- 22BCS13365 Section/Group- KPIT-901/B

Day 3

Function \$ RECURSION DSA QUESTION

Very Easy

Q .Fibonacci Series Using Recursion

The Fibonacci numbers, commonly denoted $F(n)$ form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

$$F(0) = 0, F(1) = 1$$

$$F(n) = F(n - 1) + F(n - 2), \text{ for } n > 1.$$

Given n , calculate $F(n)$.

Example 1:

Input: $n = 2$

Output: 1

Explanation: $F(2) = F(1) + F(0) = 1 + 0 = 1$

Example 2:

Input: $n = 3$

Output: 2

Explanation: $F(3) = F(2) + F(1) = 1 + 1 = 2$.

Constraints:

$0 \leq n \leq 30$

Solution:

```
#include <iostream>

using namespace std;

int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
    int n;

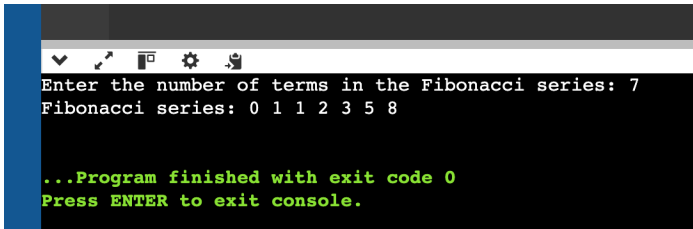
    cout << "Enter the number of terms in the Fibonacci series: ";

    cin >> n;

    cout << "Fibonacci series: ";

    for (int i = 0; i < n; i++) {
        cout << fibonacci(i) << " ";
    }
```

```
cout << endl;  
  
return 0;  
  
}
```



```
Enter the number of terms in the Fibonacci series: 7  
Fibonacci series: 0 1 1 2 3 5 8  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Easy

Power Of Three

Given an integer n , return true if it is a power of three. Otherwise, return false.

An integer n is a power of three, if there exists an integer x such that $n == 3^x$.

Example 1:

Input: $n = 27$

Output: true

Explanation: $27 = 3^3$

Example 2:

Input: $n = 0$

Output: false

Explanation: There is no x where $3^x = 0$.

Constraints:

$-2^{31} \leq n \leq 2^{31} - 1$

Follow up: Could you solve it without loops/recursion?

Solution:

```
#include <iostream>

using namespace std;

bool isPowerOfThree(int n) {

    if (n <= 0)

        return false;

    while (n % 3 == 0) {

        n /= 3;

    }

    return n == 1;

}

int main() {

    int n;

    cout << "Enter an integer: ";

    cin >> n;

    if (isPowerOfThree(n)) {

        cout << n << " is a power of three." << endl;

    } else {

        cout << n << " is not a power of three." << endl;

    }

    return 0;

}
```

```
Enter an integer: 27
27 is a power of three.

...Program finished with exit code 0
Press ENTER to exit console.
```

Medium

Q2 . Elimination Game

You have a list `arr` of all integers in the range $[1, n]$ sorted in a strictly increasing order. Apply the following algorithm on `arr`:

Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.

Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.

Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer `n`, return the last number that remains in `arr`.

Example 1:

Input: `n = 9`

Output: 6

Explanation:

`arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]`

`arr = [2, 4, 6, 8]`

`arr = [2, 6]`

`arr = [6]`

Example 2:

Input: `n = 1`

Output: 1

Constraints:

$1 \leq n \leq 109$

Solution:

```
#include <iostream>

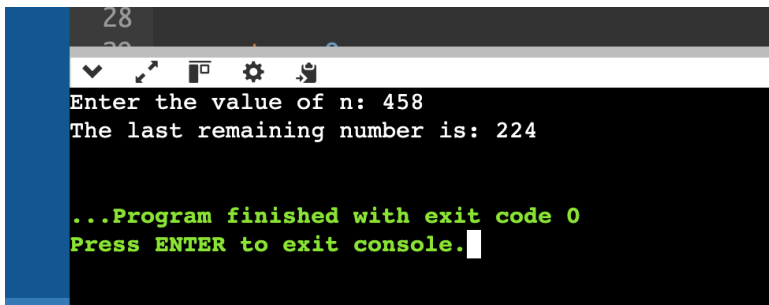
using namespace std;

int lastRemaining(int n) {
    int remaining = n;
    int step = 1;
    int head = 1;
    bool leftToRight = true;

    while (remaining > 1) {
        if (leftToRight || remaining % 2 == 1) {
            head += step;
        }
        step *= 2;
        remaining /= 2;
        leftToRight = !leftToRight;
    }

    return head;
}
```

```
int main() {  
    int n;  
    cout << "Enter the value of n: ";  
    cin >> n;  
    cout << "The last remaining number is: " << lastRemaining(n) << endl;  
    return 0;  
}
```



```
28  
29  
Enter the value of n: 458  
The last remaining number is: 224  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Hard

Q. Wildcard Matching

Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where:

'?' Matches any single character.

'*' Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

Example 1:

Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

Example 2:

Input: s = "aa", p = "*"

Output: true

Explanation: '*' matches any sequence.

Example 3:

Input: s = "cb", p = "?a"

Output: false

Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.

Constraints:

0 <= s.length, p.length <= 2000

s contains only lowercase English letters.

p contains only lowercase English letters, '?' or '*'.

Solution:

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
bool isMatch(string s, string p) {
```

```
    int m = s.size();
```

```
    int n = p.size();
```

```
    vector<vector<bool>> dp(m + 1, vector<bool>(n + 1, false));
```

```
    dp[0][0] = true;
```

```
    for (int j = 1; j <= n; j++) {
```

```
        if (p[j - 1] == '*') {
```

```
            dp[0][j] = dp[0][j - 1];
```

```
        }
```



```

}
for (int i = 1; i <= m; i++) {
    for (int j = 1; j <= n; j++) {
        if (p[j - 1] == s[i - 1] || p[j - 1] == '?') {
            dp[i][j] = dp[i - 1][j - 1];
        } else if (p[j - 1] == '*') {
            dp[i][j] = dp[i - 1][j] || dp[i][j - 1];
        }
    }
}
return dp[m][n];
}

int main() {
    string s, p;
    cin >> s >> p;
    if (isMatch(s, p)) {
        cout << "Match" << endl;
    } else {
        cout << "No Match" << endl;
    }
    return 0;
}

```



```

44
aa
*
Match

...Program finished with exit code 0
Press ENTER to exit console.

```

Very Hard

Q. Maximize Number of Nice Divisors

You are given a positive integer `primeFactors`. You are asked to construct a positive integer `n` that satisfies the following conditions:

The number of prime factors of `n` (not necessarily distinct) is at most `primeFactors`.

The number of nice divisors of `n` is maximized. Note that a divisor of `n` is nice if it is divisible by every prime factor of `n`. For example, if `n = 12`, then its prime factors are `[2,2,3]`, then 6 and 12 are nice divisors, while 3 and 4 are not.

Return the number of nice divisors of `n`. Since that number can be too large, return it modulo $10^9 + 7$.

Note that a prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. The prime factors of a number `n` is a list of prime numbers such that their product equals `n`.

Example 1:

Input: `primeFactors = 5`

Output: 6

Explanation: 200 is a valid value of `n`.

It has 5 prime factors: `[2,2,2,5,5]`, and it has 6 nice divisors: `[10,20,40,50,100,200]`.

There is not other value of `n` that has at most 5 prime factors and more nice divisors.

Example 2:

Input: `primeFactors = 8`

Output: 18

Constraints:

$1 \leq \text{primeFactors} \leq 109$

Solution:

```
#include <iostream>

using namespace std;

const int MOD = 1e9 + 7;

long long modPow(long long base, long long exp, int mod) {
    long long result = 1;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        base = (base * base) % mod;
        exp /= 2;
    }
    return result;
}

int maxNiceDivisors(int primeFactors) {
    if (primeFactors == 1) return 1;

    int a = primeFactors / 3;
    int b = primeFactors % 3;
    if (b == 0) {
```

```

    return modPow(3, a, MOD);
} else if (b == 1) {
    return (modPow(3, a - 1, MOD) * 4) % MOD;
} else { // b == 2
    return (modPow(3, a, MOD) * 2) % MOD;
}
}

int main() {
    int primeFactors;

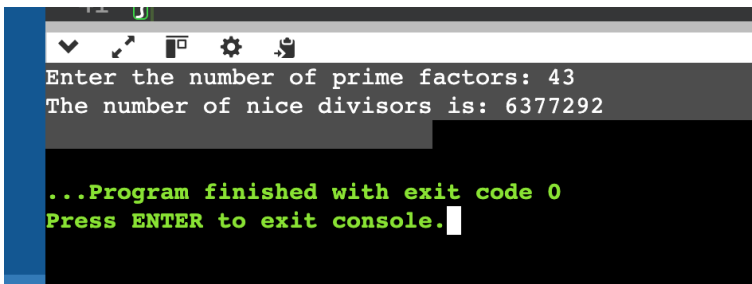
    cout << "Enter the number of prime factors: ";

    cin >> primeFactors;

    cout << "The number of nice divisors is: " << maxNiceDivisors(primeFactors) << endl;

    return 0;
}

```



```

Enter the number of prime factors: 43
The number of nice divisors is: 6377292

...Program finished with exit code 0
Press ENTER to exit console.

```