```cpp
#include <iostream>
using namespace std;

int my_pow(int value, int p ) {
    if (p == 0) {
        return 1;
    }
    return value * my_pow(value, p - 1);
}

int main() {
    int n,p;
    cin >> n >> p;
    cout << my_pow(n,p);
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int length_3n_plus_1(int n){
    if (n == 1) return 1;
    if (n % 2 == 0) return 1 + length_3n_plus_1(n / 2);
    else return 1 + length_3n_plus_1(3 * n + 1);
}

int main() {
    int n;
    cin >> n;
    cout << length_3n_plus_1(n) << endl;
    return 0;
}
```

D: > future > Yahia > task1.cpp > main()

```cpp
#include <iostream>
using namespace std;
int arr_max(int arr[], int len){
    if (len == 1) {
        return arr[0];
    }
 int max = arr_max(arr, len-1);
return (arr[len-1] > max ? arr[len-1] : max);
}

int main() {
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    cout << arr_max(arr, len) << endl;
    return 0;
}
```

D: > future > Yahia > task1.cpp > sum(int [], int)

```cpp
#include <iostream>
using namespace std;
int sum(int arr[], int len){
    if (len == 0) {
        return 0;
    }
return sum (arr, len - 1) + arr[len - 1];
}

int main() {
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    cout <<sum(arr, len) << endl;
    return 0;
}

```

```cpp
#include <iostream>
using namespace std;
double average(int arr[], int len){
  if (len == 1) {
    return arr[0];
    } else {
       double prev = average(arr, len - 1);
        return (prev * (len - 1) + arr[len - 1]) / len;
  }
}

int main() {
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    cout << average(arr, len) ;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
void array_increment(int arr[], int len, int index =0){
    if (index >= len){
     return;
    }
   arr[index] += index;
   array_increment( arr, len,  index + 1);
   }

int main() {
    int n,index;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    array_increment(arr, len, index) ;
    return 0;
}

```

```cpp
#include <iostream>
using namespace std;
void array_increment(int arr[], int len, int index =1){
    if (index >= len){
     return;
    }
   arr[index] += arr[index-1];
   array_increment( arr, len,  index + 1);
   }

int main() {
    int n,index;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    array_increment(arr, len, index) ;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
void left_max(int arr[], int len , int index = 1) {
    if (index == len) {
        return;
    }
    int max = arr[index];
    if (max > arr[index - 1]) {
        arr[index] = max;
    }
    else {
        arr[index] = arr[index - 1];
    }
    left_max(arr, len, index + 1);
  }

int main() {
    int n,index;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    left_max(arr, len);
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
void left_max(int arr[], int len ) {
    if (len == 1) {
        return;
    }
    int max = arr[len-1];
    if (max > arr[len - 2]) {
        arr[len-2] = max;
    }
    left_max(arr, len - 1);
  }

int main() {
    int n,index;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    left_max(arr, len);
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
void suffix_sum(int arr[], int len, int last_elment , int & sum) {
    if (last_elment == 0) {
        return;
    }
    sum += arr[len-last_elment] ;

    suffix_sum(arr, len, last_elment-1, sum);
  }

int main() {
    int n,index,sum = 0;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int last_elment;
    cin >> last_elment;
    int len = n;
    suffix_sum(arr, len, last_elment,sum);
    cout << sum << endl;
    return 0;
}
```

D: > future > Yahia > task1.cpp > ...

```cpp
#include <iostream>
using namespace std;
void suffix_sum(int arr[], int len, int last_elment , int & sum ,int index = 0) {
    if (index == last_elment) {
        return;
    }
    sum += arr[index] ;

    suffix_sum(arr, len, last_elment, sum, index + 1);
  }

int main() {
    int n,index,sum = 0;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int last_elment;
    cin >> last_elment;
    int len = n;
    suffix_sum(arr, len, last_elment,sum);
    cout << sum << endl;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int is_plandrom(int arr[], int len  ,int index = 0) {
    if (index >= len / 2) {
        return 1;
    }
    if (arr[index]!= arr[len-index-1] ) {
        return 0;
    }

    is_plandrom(arr, len,  index + 1);
  }

int main() {
    int n,index,sum = 0;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int len = n;
    is_plandrom(arr, len,index);
    return 0;
}
```

```cpp
#include <iostream>
#include <string>
using namespace std;

int is_prefix(string main, string prefix, int main_idx = 0, int prefix_idx = 0) {
    if (prefix_idx == prefix.length()) {
        return 1;
    }
    if (main_idx >= main.length() || main[main_idx] != prefix[prefix_idx]) {
        return 0;
    }
    return is_prefix(main, prefix, main_idx + 1, prefix_idx + 1);
}

int main() {
    string main, prefix;
    int main_idx = 0, prefix_idx = 0;

    cout << "Enter the main string: ";
    cin >> main;
    cout << "Enter the prefix string: ";
    cin >> prefix;

    cout << is_prefix(main, prefix, main_idx, prefix_idx);
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

bool is_prime(int n, int i = 2) {
    if (n <= 1) return false;
    if (i * i > n) return true;
    if (n % i == 0) return false;
    return is_prime(n, i + 1);
}
int count_primes(int start, int end) {
    if (start > end) return 0;
    return (is_prime(start) ? 1 : 0) + count_primes(start + 1, end);
}
int main() {
    int start = 10, end = 20;
    cout << "Primes between " << start << " and " << end << ": "
         << count_primes(start, end) << endl;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

int path_sum(int grid[100][100], int row, int col, int ROWS, int COLS) {
    if (row >= ROWS || col >= COLS)
        return 0;
    int right = (col + 1 < COLS) ? grid[row][col + 1] : -1;
    int down = (row + 1 < ROWS) ? grid[row + 1][col] : -1;
    int diag = (row + 1 < ROWS && col + 1 < COLS) ? grid[row + 1][col + 1] : -1;
    if (right >= down && right >= diag)
        return grid[row][col] + path_sum(grid, row, col + 1, ROWS, COLS);
    else if (down >= right && down >= diag)
        return grid[row][col] + path_sum(grid, row + 1, col, ROWS, COLS);
    else
        return grid[row][col] + path_sum(grid, row + 1, col + 1, ROWS, COLS);
}
int main() {
    int grid[100][100];
    int ROWS, COLS;
    cout << "Enter number of rows and columns: ";
    cin >> ROWS >> COLS;
    cout << "Enter the grid values:\n";
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            cin >> grid[i][j];
        }
    }
    int result = path_sum(grid, 0, 0, ROWS, COLS);
    cout << "Maximum path sum: " << result << endl;

    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int fibonacci(int n) {
    int fib[n + 1];
    fib[1] = 1;
    fib[2] = 1;
    for (int i = 3; i <= n; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
    return fib[n];
}
int main() {
    int n;
    cin >> n;
    cout << fibonacci(n+1) << endl;
    return 0;
}
```