

```
1  #include <iostream>
2  using namespace std;
3
4  int my_pow(int value, int p ) {
5      if (p == 0) {
6          return 1;
7      }
8      return value * my_pow(value, p - 1);
9  }
10
11 int main() {
12     int n,p;
13     cin >> n >> p;
14     cout << my_pow(n,p);
15     return 0;
16 }
```

```
1  #include <iostream>
2  using namespace std;
3  int length_3n_plus_1(int n){
4      if (n == 1) return 1;
5      if (n % 2 == 0) return 1 + length_3n_plus_1(n / 2);
6      else return 1 + length_3n_plus_1(3 * n + 1);
7  }
8
9  int main() {
10     int n;
11     cin >> n;
12     cout << length_3n_plus_1(n) << endl;
13     return 0;
14 }
15
```

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

```
1  #include <iostream>
2  using namespace std;
3  int arr_max(int arr[], int len){
4      if (len == 1) {
5          return arr[0];
6      }
7      int max = arr_max(arr, len-1);
8      return (arr[len-1] > max ? arr[len-1] : max);
9  }
10
11 int main() {
12     int n;
13     cin >> n;
14     int arr[n];
15     for (int i = 0; i < n; i++) {
16         cin >> arr[i];
17     }
18     int len = n;
19     cout << arr_max(arr, len) << endl;
20     return 0;
21 }
22
```

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

```
1  #include <iostream>
2  using namespace std;
3  int sum(int arr[], int len){
4      if (len == 0) {
5          return 0;
6      }
7      return sum (arr, len - 1) + arr[len - 1];
8  }
9
10 int main() {
11     int n;
12     cin >> n;
13     int arr[n];
14     for (int i = 0; i < n; i++) {
15         cin >> arr[i];
16     }
17     int len = n;
18     cout << sum(arr, len) << endl;
19     return 0;
20 }
21
```

```
1  #include <iostream>
2  using namespace std;
3  double average(int arr[], int len){
4      if (len == 1) {
5          return arr[0];
6      } else {
7          double prev = average(arr, len - 1);
8          return (prev * (len - 1) + arr[len - 1]) / len;
9      }
10 }
11
12 int main() {
13     int n;
14     cin >> n;
15     int arr[n];
16     for (int i = 0; i < n; i++) {
17         cin >> arr[i];
18     }
19     int len = n;
20     cout << average(arr, len) ;
21     return 0;
22 }
```

```
1  #include <iostream>
2  using namespace std;
3  void array_increment(int arr[], int len, int index =0){
4      if (index >= len){
5          return;
6      }
7      arr[index] += index;
8      array_increment( arr, len,  index + 1);
9  }
10
11 int main() {
12     int n,index;
13     cin >> n;
14     int arr[n];
15     for (int i = 0; i < n; i++) {
16         cin >> arr[i];
17     }
18     int len = n;
19     array_increment(arr, len, index) ;
20     return 0;
21 }
22
```


```
1  #include <iostream>
2  using namespace std;
3  void array_increment(int arr[], int len, int index =1){
4      if (index >= len){
5          return;
6      }
7      arr[index] += arr[index-1];
8      array_increment( arr, len,  index + 1);
9  }
10
11 int main() {
12     int n,index;
13     cin >> n;
14     int arr[n];
15     for (int i = 0; i < n; i++) {
16         cin >> arr[i];
17     }
18     int len = n;
19     array_increment(arr, len, index) ;
20     return 0;
21 }
```

```
1  #include <iostream>
2  using namespace std;
3  void left_max(int arr[], int len , int index = 1) {
4      if (index == len) {
5          return;
6      }
7      int max = arr[index];
8      if (max > arr[index - 1]) {
9          arr[index] = max;
10     }
11     else {
12         arr[index] = arr[index - 1];
13     }
14     left_max(arr, len, index + 1);
15 }
16
17 int main() {
18     int n,index;
19     cin >> n;
20     int arr[n];
21     for (int i = 0; i < n; i++) {
22         cin >> arr[i];
23     }
24     int len = n;
25     left_max(arr, len);
26     return 0;
27 }
28
```



```
1  #include <iostream>
2  using namespace std;
3  void left_max(int arr[], int len ) {
4      if (len == 1) {
5          return;
6      }
7      int max = arr[len-1];
8      if (max > arr[len - 2]) {
9          arr[len-2] = max;
10     }
11     left_max(arr, len - 1);
12 }
13
14 int main() {
15     int n,index;
16     cin >> n;
17     int arr[n];
18     for (int i = 0; i < n; i++) {
19         cin >> arr[i];
20     }
21     int len = n;
22     left_max(arr, len);
23     return 0;
24 }
25
```

```
1  #include <iostream>
2  using namespace std;
3  void suffix_sum(int arr[], int len, int last_element , int & sum) {
4      if (last_element == 0) {
5          return;
6      }
7      sum += arr[len-last_element] ;
8
9      suffix_sum(arr, len, last_element-1, sum);
10 }
11
12 int main() {
13     int n,index,sum = 0;
14     cin >> n;
15     int arr[n];
16     for (int i = 0; i < n; i++) {
17         cin >> arr[i];
18     }
19     int last_element;
20     cin >> last_element;
21     int len = n;
22     suffix_sum(arr, len, last_element,sum);
23     cout << sum << endl;
24     return 0;
25 }
26
```

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

```
1  #include <iostream>
2  using namespace std;
3  void suffix_sum(int arr[], int len, int last_elment , int & sum ,int index = 0) {
4      if (index == last_elment) {
5          return;
6      }
7      sum += arr[index] ;
8
9      suffix_sum(arr, len, last_elment, sum, index + 1);
10 }
11
12 int main() {
13     int n,index,sum = 0;
14     cin >> n;
15     int arr[n];
16     for (int i = 0; i < n; i++) {
17         cin >> arr[i];
18     }
19     int last_elment;
20     cin >> last_elment;
21     int len = n;
22     suffix_sum(arr, len, last_elment,sum);
23     cout << sum << endl;
24     return 0;
25 }
26
```

```
1  #include <iostream>
2  using namespace std;
3  int is_plandrom(int arr[], int len ,int index = 0) {
4      if (index >= len / 2) {
5          return 1;
6      }
7      if (arr[index] != arr[len-index-1] ) {
8          return 0;
9      }
10
11      is_plandrom(arr, len,  index + 1);
12  }
13
14  int main() {
15      int n,index,sum = 0;
16      cin >> n;
17      int arr[n];
18      for (int i = 0; i < n; i++) {
19          cin >> arr[i];
20      }
21      int len = n;
22      is_plandrom(arr, len,index);
23      return 0;
24  }
25
```

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int is_prefix(string main, string prefix, int main_idx = 0, int prefix_idx = 0) {
6      if (prefix_idx == prefix.length()) {
7          return 1;
8      }
9      if (main_idx >= main.length() || main[main_idx] != prefix[prefix_idx]) {
10         return 0;
11     }
12     return is_prefix(main, prefix, main_idx + 1, prefix_idx + 1);
13 }
14
15 int main() {
16     string main, prefix;
17     int main_idx = 0, prefix_idx = 0;
18
19     cout << "Enter the main string: ";
20     cin >> main;
21     cout << "Enter the prefix string: ";
22     cin >> prefix;
23
24     cout << is_prefix(main, prefix, main_idx, prefix_idx);
25     return 0;
26 }
27
```

```
1  #include <iostream>
2  using namespace std;
3
4  bool is_prime(int n, int i = 2) {
5      if (n <= 1) return false;
6      if (i * i > n) return true;
7      if (n % i == 0) return false;
8      return is_prime(n, i + 1);
9  }
10 int count_primes(int start, int end) {
11     if (start > end) return 0;
12     return (is_prime(start) ? 1 : 0) + count_primes(start + 1, end);
13 }
14 int main() {
15     int start = 10, end = 20;
16     cout << "Primes between " << start << " and " << end << ": "
17         << count_primes(start, end) << endl;
18     return 0;
19 }
20
```

```

1  #include <iostream>
2  using namespace std;
3
4  int path_sum(int grid[100][100], int row, int col, int ROWS, int COLS) {
5      if (row >= ROWS || col >= COLS)
6          return 0;
7      int right = (col + 1 < COLS) ? grid[row][col + 1] : -1;
8      int down = (row + 1 < ROWS) ? grid[row + 1][col] : -1;
9      int diag = (row + 1 < ROWS && col + 1 < COLS) ? grid[row + 1][col + 1] : -1;
10     if (right >= down && right >= diag)
11         return grid[row][col] + path_sum(grid, row, col + 1, ROWS, COLS);
12     else if (down >= right && down >= diag)
13         return grid[row][col] + path_sum(grid, row + 1, col, ROWS, COLS);
14     else
15         return grid[row][col] + path_sum(grid, row + 1, col + 1, ROWS, COLS);
16 }
17 int main() {
18     int grid[100][100];
19     int ROWS, COLS;
20     cout << "Enter number of rows and columns: ";
21     cin >> ROWS >> COLS;
22     cout << "Enter the grid values:\n";
23     for (int i = 0; i < ROWS; i++) {
24         for (int j = 0; j < COLS; j++) {
25             cin >> grid[i][j];
26         }
27     }
28     int result = path_sum(grid, 0, 0, ROWS, COLS);
29     cout << "Maximum path sum: " << result << endl;
30
31     return 0;
32 }
33

```

```
1  #include <iostream>
2  using namespace std;
3  int fibonacci(int n) {
4      int fib[n + 1];
5      fib[1] = 1;
6      fib[2] = 1;
7      for (int i = 3; i <= n; i++) {
8          fib[i] = fib[i - 1] + fib[i - 2];
9      }
10     return fib[n];
11 }
12 int main() {
13     int n;
14     cin >> n;
15     cout << fibonacci(n+1) << endl;
16     return 0;
17 }
18
```