Erudite Translation - Attempter Instructions

👋 Welcome to the Erudite Translation Project!

In this project, you'll be presented with a coding problem, solution, and test cases written in Python. You will be asked to convert these elements into a target language such as JavaScript, Go, Java, C++, and Rust.

The goal of this project is to create a solution to the prompt in the target language that leverages all the strengths of the language. You can use the provided Python solution as a reference, but you do not have to follow the same logic. If there is a more optimal way to approach the problem using target language-specific features, please use these features to implement it. **The Python solutions may have bugs!**

There is a video recording of a task here for your reference.

Pick the programming language you'd like to target into by filling out this form: Erudite Translation - Skills Form

You can only choose one at a time, but no worries! You can always **switch later using the same form**. For example, if you change your mind or if the language you picked runs out of tasks, you're free to pick another one you know.

Just keep in mind that you should be an expert in the language you select, as the quality of your work will depend on it.

LLM use is strictly prohibited on this project! There are serious legal implications to LLMs being used to generate training data. We will be checking aggressively for LLM use, which is grounds for removal from the platform.

Project Rules

Do Not:
- ✘ Task without reading this guide.
- ✘ Use an LLM to complete the task.

Do:
- ✔ Take your time and focus on details.
- ✔ Refer back to this guide whenever you task.

🔄 Change Log

Date	Updates	Notes/Images/Links
Oct 5, 2025	Updated Rust template	
Sep 30, 2025	Added common errors	

Sep 24, 2025	Initial draft

📋 Steps Overview

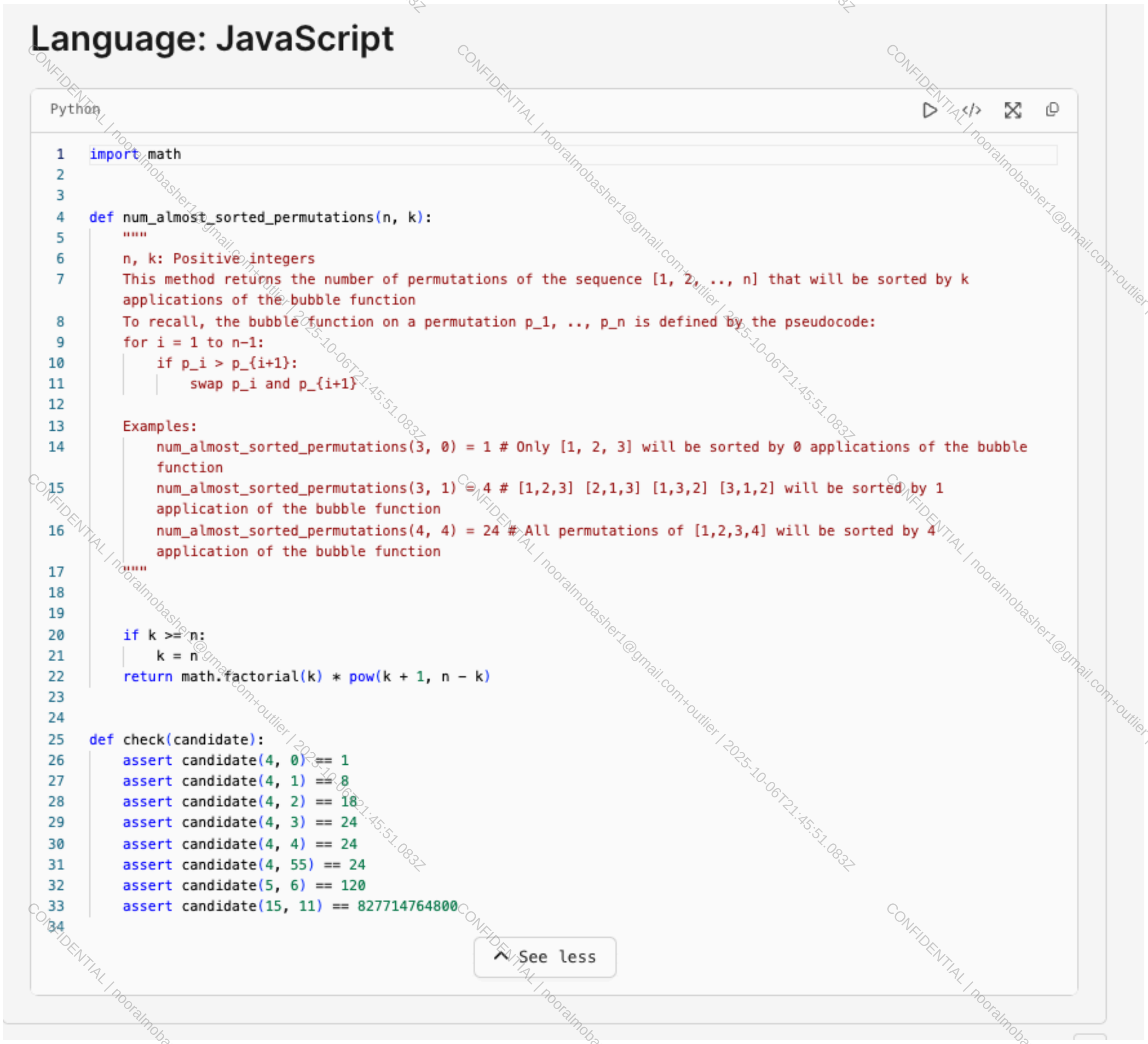
This is a high-level overview of the four phases of a task.

Step	Title	Description
1	STEP 1: **Understand the Problem**	**Analyze** the original Python prompt, reference solution, and test cases to fully grasp the core logic, constraints, and expected outcomes
2	STEP 2: **Make a Game Plan**	**Design a new solution** that is idiomatic to the target language. Plan to use language-specific features, data structures, and best practices to **achieve the most efficient and readable code**
3	STEP 3: **Code It & Check It**	**Write the solution** and target the test cases into the target language. ● Compile and run the tests to verify correctness ● Handle edge cases ● Ensure your solution functions as expected
4	STEP 4: **Polish & Send It**	**Review** your implementation for **code quality, style, and efficiency**. Once polished, prepare the final prompt, solution, and test case files in the target language for submission

🪜 Step-by-Step Breakdown

Step 1: Read the Python Solution Files

Carefully read the Python prompt, solution, and test files. These will be present in the Outlier UI as well as in the coding environment. If you are not comfortable with the target language, please skip the task and update your specialization.

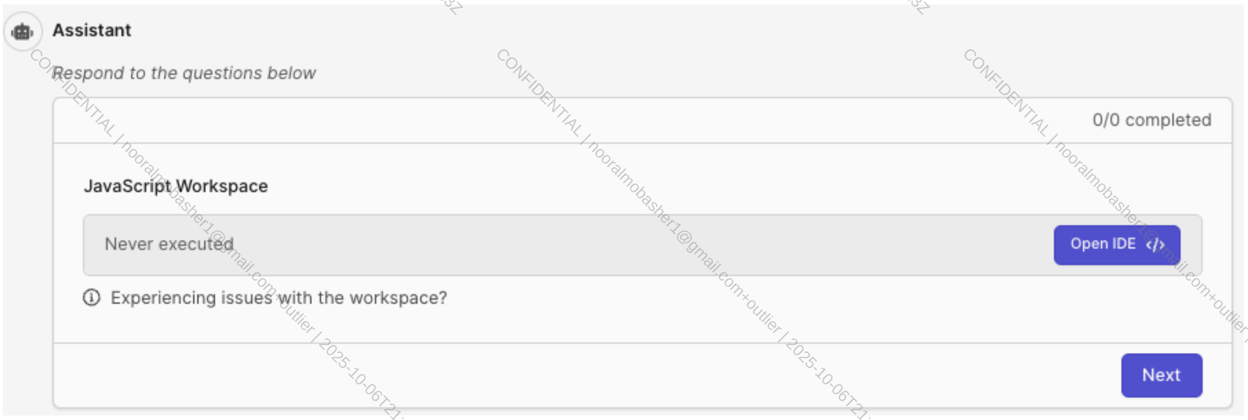

Language: JavaScript

Python

```python
import math


def num_almost_sorted_permutations(n, k):
    """
    n, k: Positive integers
    This method returns the number of permutations of the sequence [1, 2, .., n] that will be sorted by k
    applications of the bubble function
    To recall, the bubble function on a permutation p_1, .., p_n is defined by the pseudocode:
    for i = 1 to n-1:
        if p_i > p_{i+1}:
            swap p_i and p_{i+1}

    Examples:
        num_almost_sorted_permutations(3, 0) = 1 # Only [1, 2, 3] will be sorted by 0 applications of the bubble
        function
        num_almost_sorted_permutations(3, 1) = 4 # [1,2,3] [2,1,3] [1,3,2] [3,1,2] will be sorted by 1
        application of the bubble function
        num_almost_sorted_permutations(4, 4) = 24 # All permutations of [1,2,3,4] will be sorted by 4
        application of the bubble function
    """


    if k >= n:
        k = n
    return math.factorial(k) * pow(k + 1, n - k)


def check(candidate):
    assert candidate(4, 0) == 1
    assert candidate(4, 1) == 8
    assert candidate(4, 2) == 18
    assert candidate(4, 3) == 24
    assert candidate(4, 4) == 24
    assert candidate(4, 55) == 24
    assert candidate(5, 6) == 120
    assert candidate(15, 11) == 827714764800

```

Step 2: Open Sphere Engine

In the task, you will find the Sphere Engine:



Assistant

Respond to the questions below

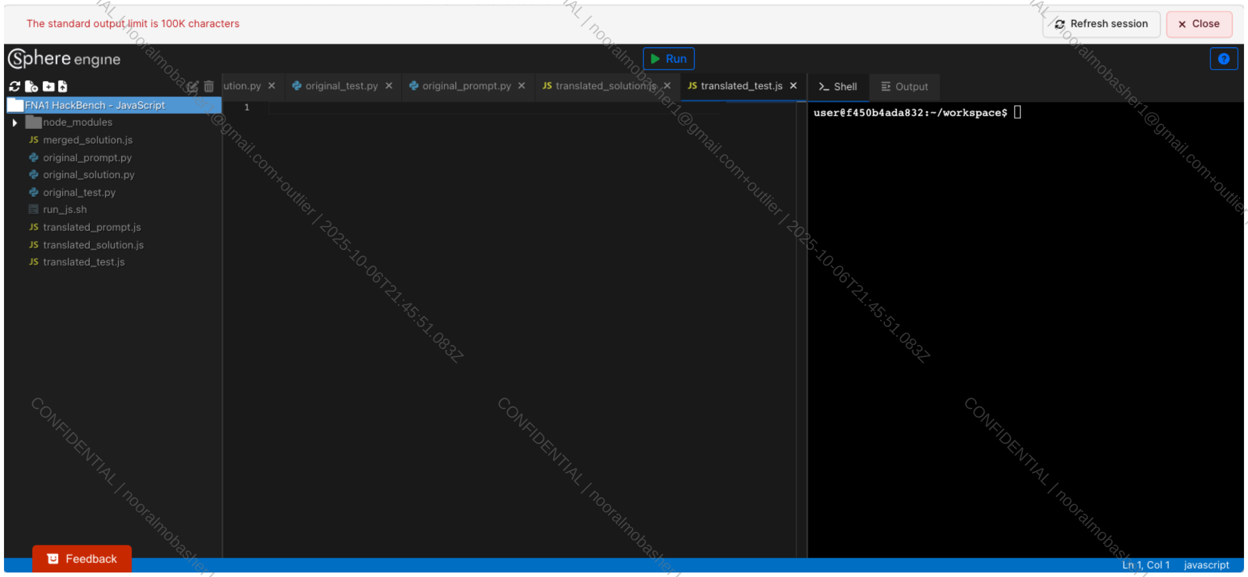
0/0 completed

JavaScript Workspace

Never executed

Experiencing issues with the workspace?

In this section, you have to click on the "Open IDE </>" button, once you click on that button, the IDE will open and you will see this:



The standard output limit is 100K characters

```
FNA1 HackBench - JavaScript
  node_modules
  merged_solution.js
  original_prompt.py
  original_solution.py
  original_test.py
  run_js.sh
  translated_prompt.js
  translated_solution.js
  translated_test.js
```

```
user@f450b4ada832:~/workspace$
```

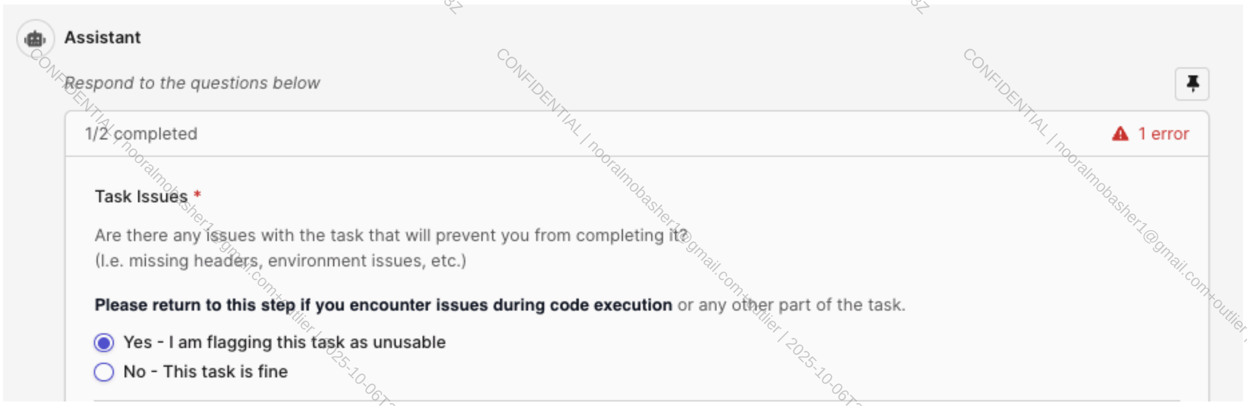
From here, you can run the code using the Run button at the top of the UI. These will trigger the run_{language}.sh shell script which does the following:

1. Performs any necessary compilations or builds
2. Runs the code

You are allowed to modify the run shell script if needed to fit the language you are working on. If you have any feedback on the structure, please submit it through the Outlier Community.

Note that in order to complete the task, the Run command must execute successfully. You can also manually trigger the run_tests.sh using the instructions from the appendix.

If you find any technical problems, please flag the task with the following field:


Assistant

Respond to the questions below

1/2 completed ⚠ 1 error

Task Issues *

Are there any issues with the task that will prevent you from completing it?
(I.e. missing headers, environment issues, etc.)

Please return to this step if you encounter issues during code execution or any other part of the task.

[✓ Yes - I am flagging this task as unusable]
No - This task is fine

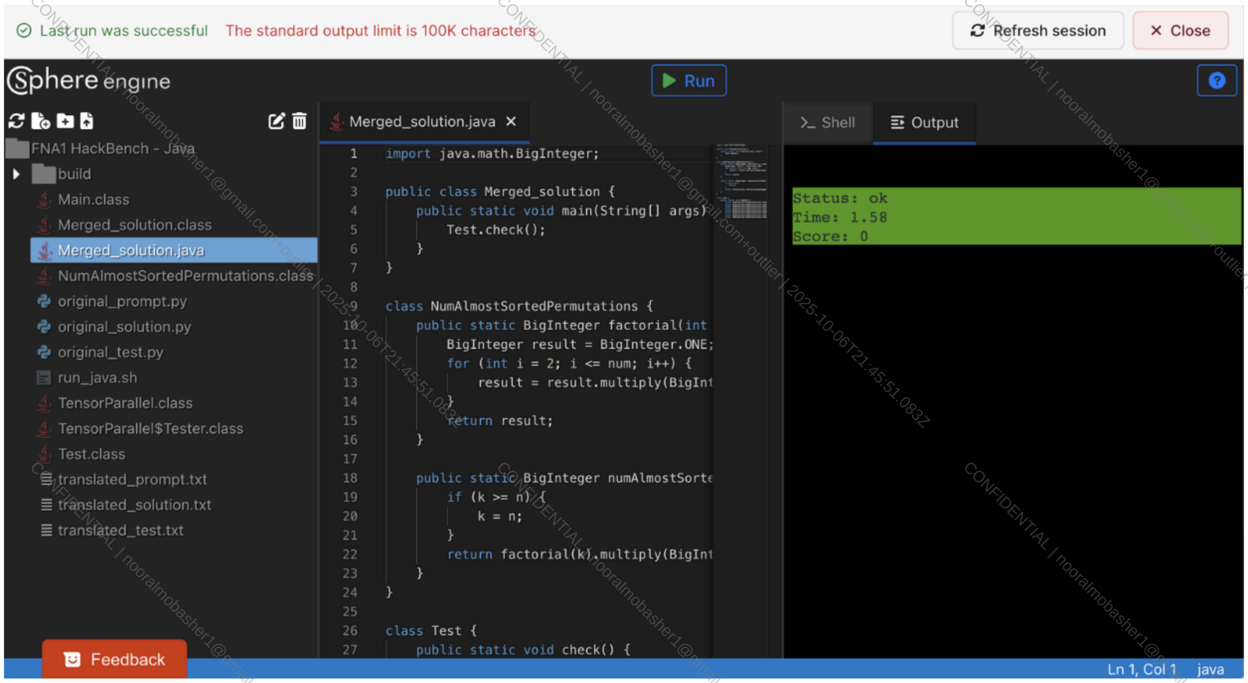
Step 3: Target the Code

For each task in each language, there are 4 important target files:

- target_prompt
- target_solution
- target_test
- merged_solution

Each target_xxx file should be the equvalent function of the original_xxx file in Python. target_test.rs should be the equivalent function of original_test.py for example.

To execute the code please compile each element into a single merged_solution file that can run your solution against the tests. For Java you will use a single class file.



```
import java.math.BigInteger;

public class Merged_solution {
    public static void main(String[] args)
        Test.check();
    }
}

class NumAlmostSortedPermutations {
    public static BigInteger factorial(int
        BigInteger result = BigInteger.ONE;
        for (int i = 2; i <= num; i++) {
            result = result.multiply(BigInt
        }
        return result;
    }

    public static BigInteger numAlmostSorte
        if (k >= n) {
            k = n;
        }
        return factorial(k).multiply(BigInt
    }
}

class Test {
    public static void check() {
```

```
Status: ok
Time: 1.58
Score: 0
```

Target Prompt Guidelines

- **Imports:** Imports (i.e. #include in C++) should only be in the target prompt file. No imports in the target solution or test files.
- **Docstrings:** The Python docstring should be converted to the equivalent docstring in the target language, according to best practices (/// for Rust etc.)
    - Note that all variable names, code snippets etc. should be converted to the target language (snake_case vs. camelCase)
- **Formatting:** Each language will have its own specific format for the function/class “stub” or “skeleton”. Ensure this is included correctly.

The examples section contains more information on each.

Target Solution Guidelines

The goal of this project is creating a solution to the prompt in the target language that leverages all strengths of the language. You can use the provided Python solution as a reference, but you do not have to follow the same logic. If there is a more optimal way to approach the problem using target language specific features, please use these features to implement it.

You must consider edge cases! Ensure that the Python and target code perform the same way. If there are bugs in the Python code you should not replicate the bugs in your code!

Dependencies
You can use any standard external dependencies that are considered best practices. If you are unable to add these to Sphere yourself, please mark the task as unusable and we will add the dependency for you.

Target Test Guidelines

The tests should be the idiomatic equivalent of the Python tests (See examples for more detail). You can tweak this based on your specific problem provided the tests have the same function as in Python.

- Python - assert candidate({inputs}) == {expected_outputs}
- JavaScript - if (candidate({inputs}) !== {expected_outputs}) throw new Error('{Message}');
- Go - if !reflect.DeepEqual(candidate({inputs}, {expected_outputs}) {panic("{Message}")}
- Java - assert {Class}.{method}({inputs}).equals({expectedOutputs});
- C++ - assert(candidate({inputs}) == {expected_outputs});
- Rust - assert_eq!(candidate({inputs}), {expected_outputs});

If there are edge cases or valuable tests missing please add them!

Understanding Entry Points

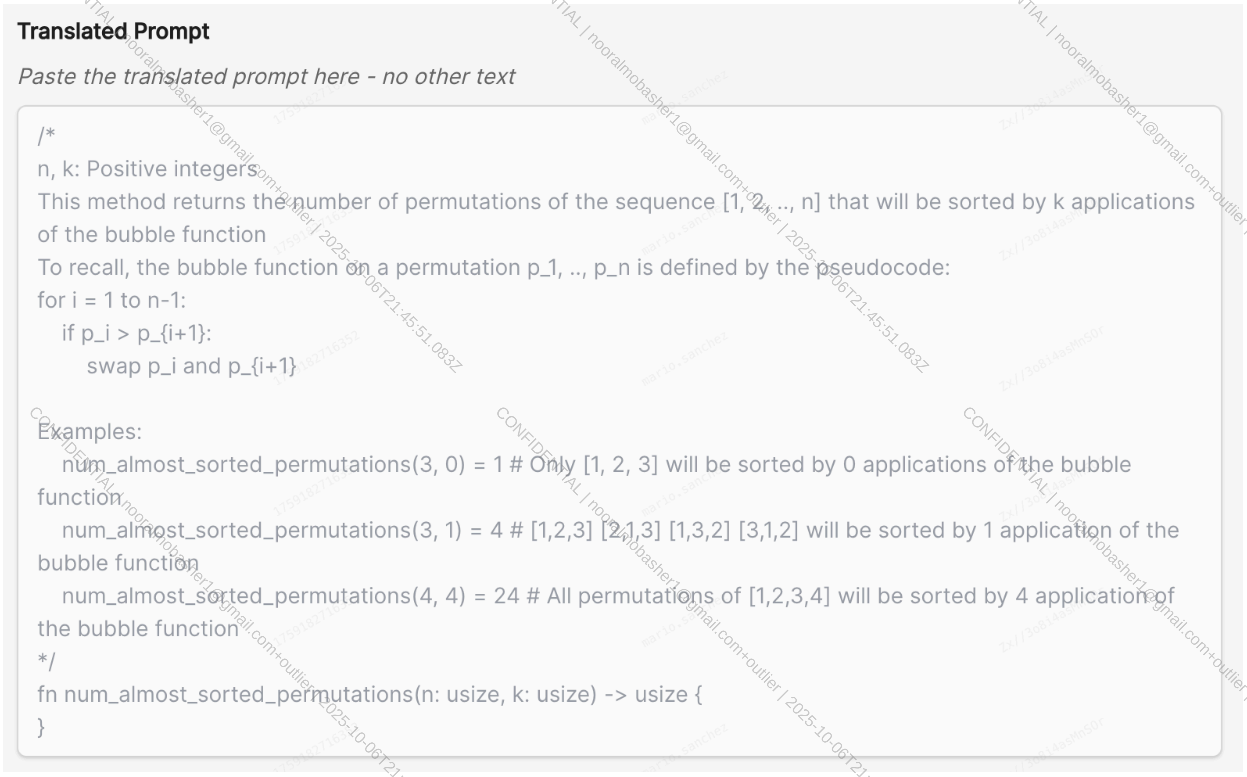
The entry point is meant to be the part of the code that gets called to solve a given input.
Entry point

- JavaScript - Function name of solution function
- Go - Function name of solution function
- Java - Class.method of the solution
- C++ - Function name of solution function
- Rust - Function name of solution function

Step 4: Paste your Solutions

Once your code executes successfully, you can exit the environment and paste the content of target_prompt, target_solution, and target_test into the provided Outlier fields.

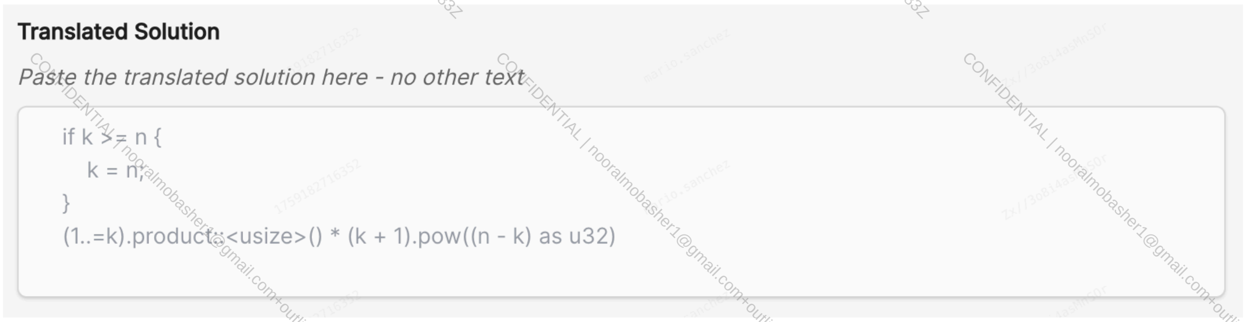
These are the fields that will be graded. Please make sure that they 100% match your final solution.



Translated Prompt

Paste the translated prompt here - no other text

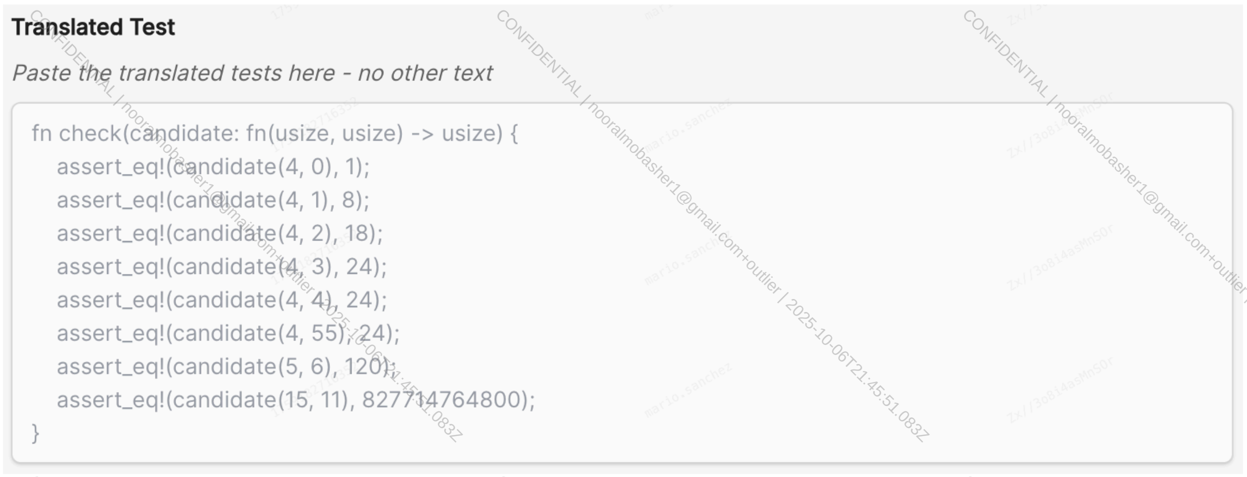
```
/*
n, k: Positive integers
This method returns the number of permutations of the sequence [1, 2, .., n] that will be sorted by k applications
of the bubble function
To recall, the bubble function on a permutation p_1, .., p_n is defined by the pseudocode:
for i = 1 to n-1:
    if p_i > p_{i+1}:
        swap p_i and p_{i+1}

Examples:
    num_almost_sorted_permutations(3, 0) = 1 # Only [1, 2, 3] will be sorted by 0 applications of the bubble
function
    num_almost_sorted_permutations(3, 1) = 4 # [1,2,3] [2,1,3] [1,3,2] [3,1,2] will be sorted by 1 application of the
bubble function
    num_almost_sorted_permutations(4, 4) = 24 # All permutations of [1,2,3,4] will be sorted by 4 application of
the bubble function
*/
fn num_almost_sorted_permutations(n: usize, k: usize) -> usize {
}
```



Translated Solution

Paste the translated solution here - no other text

```
    if k >= n {
        k = n;
    }
    (1..=k).product::<usize>() * (k + 1).pow((n - k) as u32)
```



Translated Test

Paste the translated tests here - no other text

```
fn check(candidate: fn(usize, usize) -> usize) {
    assert_eq!(candidate(4, 0), 1);
    assert_eq!(candidate(4, 1), 8);
    assert_eq!(candidate(4, 2), 18);
    assert_eq!(candidate(4, 3), 24);
    assert_eq!(candidate(4, 4), 24);
    assert_eq!(candidate(4, 55), 24);
    assert_eq!(candidate(5, 6), 120);
    assert_eq!(candidate(15, 11), 827714764800);
}
```

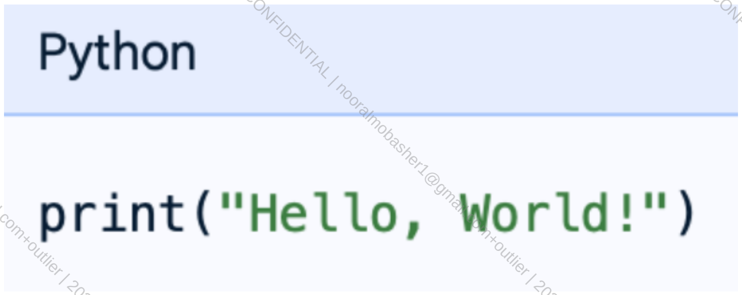
Step 5: Submit the Task

Submit the task!

⏱️ Please note that if you are flagging a task as unusable or saying no changes are needed your task time should be much shorter! We will be checking this.

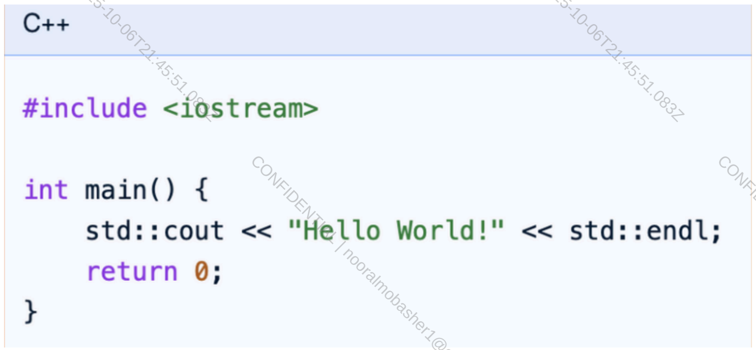
Keep in mind:

Even if you target the Python file literally, you have to **take into account best practices and everything else involved in the other language**. For example:



Python

```python
print("Hello, World!")
```



C++

```cpp
#include <iostream>

int main() {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

The syntax does not have to be a literal target, it must **follow the rules of the language** you are target.

❌ Common Errors and How to Avoid Them

📝 Examples

Python

This is an example Python solution that will be target into each of the target languages.

Prompt

```python
import package # Just an example, no function
def fibonacci_number(n):
    """Calculates the nth Fibonacci number.

    The sequence starts with 0 and 1,where each subsequent number
    is the sum of the two preceding ones.

    Inputs:
    - n: int The index of the Fibonacci number to calculate. Must be
non-negative.
         If n < 0, the function returns 0.

    Returns:
    - int: The value of the nth Fibonnaci number.

    >>> fibonacci_number(0)
    0

    >>> fibonacci_number(1)
    1

    >>> fibonacci_number(6)
    8

    >>> fibonacci_number(10)
    55
    """
```

Solution

Python
```python
if n <= 1:
    # Handles negative input by returning 0.
    return max(0, n)

# Initialize the first two numbers
a, b = 0, 1

# Iterate from the 2nd number up to the nth number (inclusive)
for _ in range(2, n + 1):
    a, b = b, a + b

# 'a' holds the (n-1)th number, 'b' holds the nth number.
return b
```

Test

Python
```python
def check(candidate):
    assert candidate(0) == 0
    assert candidate(6) == 8
    assert candidate(10) == 55
    assert candidate(12) == 144
    assert candidate(-5) == 0

    print("All assertions passed!")
```

Merged Solution

Python
```
{prompt}

{solution}

{test}
```

```
if __name__ == '__main__':
    check(fibonacci_number)
```

JavaScript

Prompt

JavaScript
```
import * from 'package'; // Just an example, no function
/**
 * Calculates the nth Fibonacci number.
 *
 * The sequence starts with 0 and 1, where each subsequent number
 * is the sum of the two preceding ones.
 *
 * @param {number} n The index of the Fibonacci number to calculate. Must be
non-negative.
 * If n < 0, the function returns 0.
 * @returns {number} The value of the nth Fibonacci number.
 *
 * @example
 * fibonacciNumber(0) // returns 0
 * fibonacciNumber(1) // returns 1
 * fibonacciNumber(6) // returns 8
 * fibonacciNumber(10) // returns 55
 */
function fibonacciNumber(n) => {
};
```

Solution

JavaScript
```
 if (n <= 1) {
    // Handles negative input by returning 0.
    return Math.max(0, n);
  }
  // Initialize the first two numbers
  let a = 0;
  let b = 1;

  // Iterate from the 2nd number up to the nth number (inclusive)
  for (let i = 2; i <= n; i++) {
    const temp = a + b;
    a = b;
    b = temp;
  }

  // 'a' holds the (n-1)th number, 'b' holds the nth number.
  return b;
```

Test

JavaScript
```javascript
function check(candidate) {
  console.assert(candidate(0) === 0, "Assertion Failed: check(0)");
  console.assert(candidate(6) === 8, "Assertion Failed: check(6)");
  console.assert(candidate(10) === 55, "Assertion Failed: check(10)");
  console.assert(candidate(12) === 144, "Assertion Failed: check(12)");
  console.assert(candidate(-5) === 0, "Assertion Failed: check(-5)");

  console.log("All assertions passed!");
}
```

Merged Solution

JavaScript
```javascript
{prompt.imports}

{solution.helper_functions}

{prompt} {
  {solution}
}

{test}

check(fibonacci_number);
```

Go

Prompt

```go
Go
package main

import ( // Just an example, no function
        "package"
        "reflect"
        "fmt"
)

// fibonacciNumber calculates the nth Fibonacci number.
//
// The sequence starts with 0 and 1, where each subsequent number
// is the sum of the two preceding ones.
//
// Inputs:
// - n: The index of the Fibonacci number to calculate. Must be non-negative.
// If n < 0, the function returns 0.
//
// Returns:
// - int: The value of the nth Fibonnaci number.
//
// Examples:
// fibonacciNumber(0) returns 0
// fibonacciNumber(1) returns 1
// fibonacciNumber(6) returns 8
// fibonacciNumber(10) returns 55
func fibonacciNumber(n int) int {

}
```

Solution

```
Go
if n <= 1 {
	// Handles negative input by returning 0.
	return int(math.Max(0, float64(n)))
}

// Initialize the first two numbers
a, b := 0, 1

// Iterate from the 2nd number up to the nth number (inclusive)
for i := 2; i <= n; i++ {
	a, b = b, a+b
}

// 'a' holds the (n-1)th number, 'b' holds the nth number.
return b
```

Test

```
Go
func check(candidate func(int, int) int64) {
	result1 := candidate(0)
	if result1 != 0 {
		panic(fmt.Printf("Test 1 failed"))
	}
	result1 := candidate(6)
	if result2 != 8 {
		panic(fmt.Printf("Test 2 failed"))
	}
	result1 := candidate(10)
	if result3 != 55 {
		panic(fmt.Printf("Test 3 failed"))
	}
	result1 := candidate(12)
	if result4 != 144 {
		panic(fmt.Printf("Test 4 failed"))
	}
	result1 := candidate(-5)
	if result5 != 0 {
```

```
                    panic(fmt.Printf("Test 5 failed"))
        }
}
```

Merged Solution

```go
{prompt.imports}

{solution.helper_functions}

{prompt} {
  {solution}
}

{test}

func main() {
       check(fibonacciNumber)
}
```

Java

Prompt

```java
import package; // Just an example, no function

public class Fibonacci {
   /**
     * Calculates the nth Fibonacci number.
     * <p>
     * The sequence starts with 0 and 1, where each subsequent number
     * is the sum of the two preceding ones.
     *
     * <p><b>Examples:</b>
     * <pre>{@code
     * fibonacciNumber(0)  // returns 0
     * fibonacciNumber(1)  // returns 1
     * fibonacciNumber(6)  // returns 8
     * fibonacciNumber(10) // returns 55
     * }</pre>
     *
     * @param n The index of the Fibonacci number to calculate. Must be
non-negative.
     * @return The value of the nth Fibonacci number. Returns 0 if n is
negative.
     */
}
```

Solution

Java
```java
public static int fibonacciNumber(int n) {
    if (n <= 1) {
        // Handles negative input by returning 0, similar to max(0, n).
        return Math.max(0, n);
    }

    // Initialize the first two numbers
    int a = 0;
    int b = 1;

    // Iterate from the 2nd number up to the nth number
    for (int i = 2; i <= n; i++) {
        int temp = a + b;
        a = b;
        b = temp;
    }

    // 'b' now holds the nth Fibonacci number
    return b;
}
```

Test

```java
class Test {
    public static void check() {
        assert fibonacciNumber(0) == 0;
        assert fibonacciNumber(6) == 8;
        assert fibonacciNumber(10) == 55;
        assert fibonacciNumber(12) == 144;
        assert fibonacciNumber(-5) == 0;

        System.out.println("All assertions passed!");
    }
}
```

Merged Solution

```java
{prompt} {
  {solution}

  {test}

  public static void main(String[] args) {
       Test.check();
  }
}
```

C++

Prompt

```cpp
#include <package> // Just an example, no function
#include <iostream>
#include <cassert>
#include <functional>
#include <algorithm>

/**
```

```
 * @brief Calculates the nth Fibonacci number.
 *
 * The sequence starts with 0 and 1, where each subsequent number
 * is the sum of the two preceding ones.
 *
 * @param n The index of the Fibonacci number to calculate. Must be
non-negative.
 * If n < 0, the function returns 0.
 * @return The value of the nth Fibonacci number.
 *
 * @b Examples:
 * @code
 * fibonacci_number(0)  // Returns 0
 * fibonacci_number(1)  // Returns 1
 * fibonacci_number(6)  // Returns 8
 * fibonacci_number(10) // Returns 55
 * @endcode
 */
int fibonacci_number(int n) {

}
```

Solution

C/C++
```
if (n <= 1) {
    // Handles negative input by returning 0, and base cases 0 and 1.
    return std::max(0, n);
}

// Initialize the first two numbers.
int a = 0;
int b = 1;

// Iterate from the 2nd number up to the nth number.
for (int i = 2; i <= n; ++i) {
    int next_fib = a + b;
    a = b;
    b = next_fib;
}
```

```
// 'a' holds the (n-1)th number, 'b' holds the nth number.
return b;
```

Test

C/C++
```
void check(const std::function<int(int)>& candidate) {
    assert(candidate(0) == 0);
    assert(candidate(6) == 8);
    assert(candidate(10) == 55);
    assert(candidate(12) == 144);
    assert(candidate(-5) == 0);

    std::cout << "All assertions passed!" << std::endl;
}
```

Merged Solution

C/C++
```
{prompt.includes}

{solution.helper_functions}

{prompt} {
  {solution}
}

{test}

int main() {
    check(fibonacci_number);
    return 0;
}
```

Rust

Prompt

Rust
```rust
use package::Package; // Just an example, no function

/// Calculates the nth Fibonacci number.
///
/// The sequence starts with 0 and 1, where each subsequent number
/// is the sum of the two preceding ones.
///
/// # Arguments
///
/// * `n`: An `i32` representing the index of the Fibonacci number to
calculate.
///       If n is negative or zero, the function returns 0.
///
/// # Returns
///
/// * A `u64` containing the value of the nth Fibonacci number.
///
/// # Examples
///
/// ```
/// // These examples are run as tests when you execute `cargo test`.
/// assert_eq!(fibonacci_number(0), 0);
/// assert_eq!(fibonacci_number(1), 1);
/// assert_eq!(fibonacci_number(6), 8);
/// assert_eq!(fibonacci_number(10), 55);
/// ```
fn fibonacci_number(n: i32) -> u64 {

}
```

Solution

Rust
```rust
// Handles negative input and the base case for n=0.
if n <= 0 {
    return 0;
}
```

```rust
// Handles the base case for n=1.
if n == 1 {
    return 1;
}

let mut a: u64 = 0;
let mut b: u64 = 1;

// Iterate from the 2nd number up to the nth number (inclusive).
for _ in 2..=n {
    (a, b) = (b, a + b);
}

// 'a' holds the (n-1)th number, 'b' holds the nth number.
b
```

Test

Rust

```rust
#[cfg(test)]
mod tests {
    // Import the function from the parent module.
    use super::*;

    #[test]
    fn test_fibonacci_cases() {
        // assert_eq! is a macro that checks for equality.
        assert_eq!(fibonacci_number(0), 0);
        assert_eq!(fibonacci_number(6), 8);
        assert_eq!(fibonacci_number(10), 55);
        assert_eq!(fibonacci_number(12), 144);
        assert_eq!(fibonacci_number(-5), 0);

        // The test runner will automatically print a success message
        // if all assertions pass.
        println!("All assertions passed!");
    }
}
```

Merged Solution

Rust

```rust
{prompt} {
  {solution}
}

{test}
```