**Instructions**

Close

**Lab instructions**

0 of 1 completed

Introduction to Recursion with Fibonacci Sequence Example Lab

​

**Introduction:**

Recursion is a fundamental concept in computer science and programming where a function calls itself to solve a problem. It is particularly useful for problems that can be broken down into smaller, similar sub-problems, such as calculating the Fibonacci sequence.

In this lab, you'll learn how to write a recursive solution to compute the Fibonacci sequence. You'll explore the core concepts of recursion, including the **base case** and **recursive case**, and understand how to combine them effectively to solve problems.

**Goal:**

The goal of this lab is to understand and implement a recursive function that calculates the n-th Fibonacci number using base and recursive cases.

**Objectives:**

* Understand the Fibonacci sequence and its recursive definition.
* Learn the difference between base cases and recursive cases.
* Implement a recursive function to compute Fibonacci numbers.
* Analyse and discuss the efficiency of the recursive solution.

**Learner Instructions:**

**Part 1: Understanding the Problem**

**Step 1**: Open the **recursive\_fibonacci.js** file located in the **PROJECT** folder. This file contains an initial implementation of the Fibonacci function and placeholders for further work.

**Step 2**: Review the comments provided in the file. These comments explain the purpose of the Fibonacci sequence, the concept of recursion, and the difference between base and recursive cases.

**Part 2: Writing the Recursive Function**

**Step 1**: Understand the base case and the recursive case.

**1. Base Case**:

The base case is the condition in a recursive function that stops the recursion. It represents the simplest, where the answer is already known and can be returned without making further recursive calls.

For Fibonacci, the base cases are:

* fib(0)=0 : When n=0, the Fibonacci number is 0.
* fib(1)=1 : When n=1, the Fibonacci number is 1.

**2. Recursive Case**:

The recursive case is the part of the function that defines how the problem is broken down into smaller, similar sub-problems. It involves the function calling itself with smaller or reduced input, moving closer to the base case with each call. This ensures the problem is eventually solved through a series of smaller, manageable steps.

For Fibonacci, the recursive case follows the formula :

* fib(n) = fib(n-1) + fib(n-2)
* This means to calculate fib(n), you need the sum of the two previous Fibonacci numbers (fib(n−1) and fib(n−2)).

**Step 2:** Examine the Skeleton Code; open the **recursive\_fibonacci.js** file, where you’ll find a partially written function:

**Step 3:** Write the base cases.

* Replace the // Step 1 comment with the base cases:

**Step 4:** Write the recursive case.

* Replace the // Step 2 comment with the recursive logic:

**Step 5:** Complete the function.

* The completed function should look like this:

**Note:** Before looking the given solution, it is highly recommended to try writing the code yourself based on the hints that are provided and concepts you have learned.

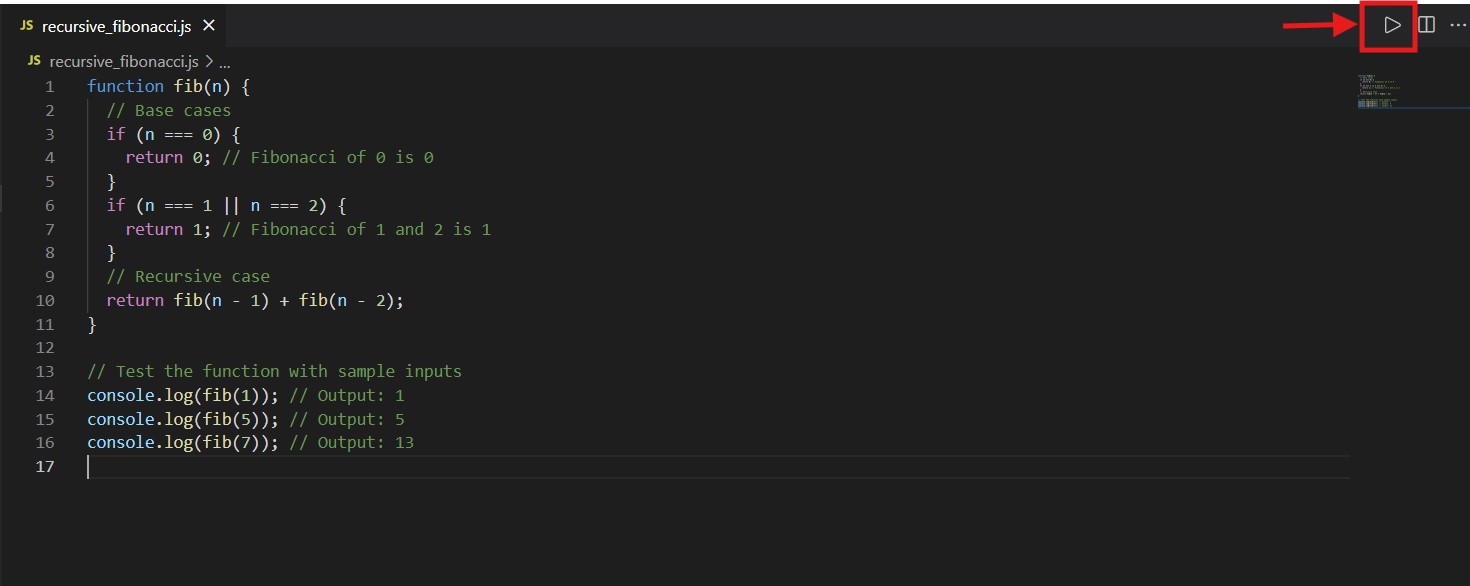
**Step 6:** Test your function.

* Add some test cases to check if your function works as expected. For example:

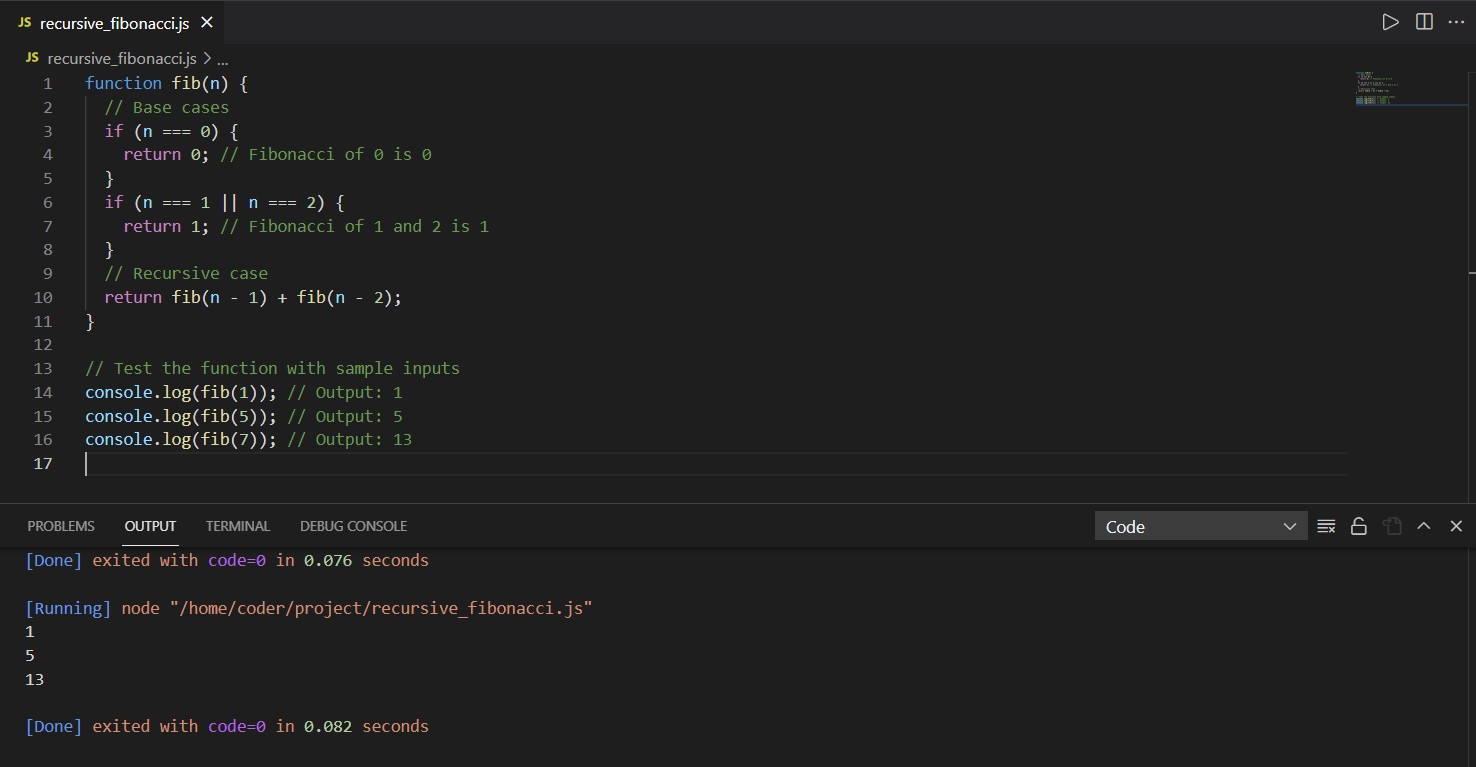
**Part 3**: **Save and Execute**

**Step 1**: Save your changes in **recursive\_fibonacci.js**.

**Step 2**: Execute the program by clicking the play button displayed below.



**Step 3:** After running this, the output will appear as shown below.



**Key Takeaways:**

* **Recursion** involves breaking a problem into smaller sub-problems.
* **Base cases** prevent infinite recursion and provide a stopping condition.
* **Recursive cases** define the problem in terms of smaller instances of itself.

**Final Step: Mark as Completed**

* Click the **Mark as Completed** button present below to mark the lab as **Completed**.