Solution Review: Calculate nth Fibonacci Number Using Recursion

Let's go over the solution review of the challenge given in the previous lesson.

We'll cover the following

Solution

Press the **RUN** button and see the output!

```
#include <iostream>

using namespace std;

// Recursive fibonacci function
int Fibonacci(int n) {

  // Base Case
  if (n == 0) {
    return 0;
  }
  else if (n == 1) {
    return 1;
  }

  // Recursive Case
  else {
    return Fibonacci(n - 1) + Fibonacci(n - 2);
  }

}

// main function
int main() {
  // Initialize variable n
  int n = 4;
  // Declare variable result
  int result;
  // Call fibonacci function in main and store its output in result
  result = Fibonacci(4);
  // Print value of result
  cout << n << "th Fibonacci number = " << result;
  return 0;
}
```

Explanation

fibonacci function

The recursive `Fibonacci` function takes a value of type `int` in its input parameters and returns the Fibonacci number at that value in the output.

Recursive case

Each element in fibonacci is a sum of its previous two elements. We recursively sum the last two elements until the base case. `fibonacci` returns the sum of `fibonacci (n-1)` + `fibonacci (n-2)`. This is the recursive case.

First base case

As the fibonacci 1st element is 1, if `n = 0`, the function terminates after returning **0** to the calling function.

Second base case

As the fibonacci 1st element is 1, if `n = 1`, the function terminates after returning **1** to the calling function.

Let's run our code for `n = 4` and see what happens inside the recursive `fibonacci` function.

> In the Fibonacci function, there are two recursive calls in the function body. Therefore, it is known as a binary recursion.

Let’s wrap up this chapter by completing a quiz in the upcoming lesson.