Activity 1: Recursive `array_max`

Like you did for the section 0 mini assignment, draw out the call stack for each recursion of `array_max([5, 1, 9, 2], 4)`. When you reach the base case and the function returns, write the return value. Continue to write the return value as you pop calls off the stack. Put "N/A" for the non-base-case “return:” values. The first one is done for you!


```
# Returns the maximum value of the first n elements in the array
# Example: array_max([5,1,9,2], 4) → 9

def array_max(array, n):
    if n == 1:
        return array[0]
    else:
        return max(array[n-1], array_max(array, n-1))
```

Activity 2: Induction Proof

Follow along with the induction proof example shown in class (also shown here on the right-hand side of the page in italics) and fill in your own proof on the left-hand side. Prove the following for all positive integers \( n \):

**Your Proof:**

\[
P(n) = \sum_{x=1}^{n} \frac{x(n+1)}{2}
\]

**Sample Proof**

The solution for the recurrence relation
\[
T(1) = c_0, T(n) = c_1 + T(n-1)
\]
is
\[
T(n) = (n-1)c_1 + c_0
\]

**Base case (show that \( P(n) \) is true for \( n = 1 \)):**

\[
T(1) = (1-1)c_1 + c_0 = c_0
\]

**Inductive Assumption (write out \( P(k) \)):**

Assume the proposition is true for \( k \):

\[
T(k) = (k-1)c_1 + c_0
\]

**Now, write out \( P(k + 1) \), what you want to prove:**

\[
T(k + 1) = (k)c_1 + c_0
\]
Inductive Step: Show that \(P(k + 1)\) is true given \(P(k)\)

Hint: Start with writing out the left side of \(P(k + 1)\) and filling in the right side based on the definition of a sum. Using simplification and your inductive assumption, make the right side look like \(P(k + 1)\).

\[
T(k + 1) = c1 + T(k) \quad \text{(by recurrence relation)}
\]

\[
T(k + 1) = c1 + (k - 1)c1 + c0 \quad \text{(by induct. assump.)}
\]

\[
T(k + 1) = (k)c1 + c0
\]

This is \(P(k + 1)\).

Conclusion (how does your work show that the claim is true?)

We’ve proven \(P(n)\) for the base case \(n = 1\) and shown that for some \(k\), \(P(k)\) implies \(P(k + 1)\), therefore \(T(n) = (n - 1)c1 + c0\) for positive integers \(n\).

Activity 3: Recursive Fibonacci

```python
function fib(n):
    if n = 0:
        return 0
    if n = 1:
        return 1
    return fib(n-1) + fib(n-2)
```

Recurrence relation:

\[
T(n) = c1 + T(n - 1) + T(n - 2)
\]

\[
T(0) = c0
\]

\[
T(1) = c0
\]

\[
T(2) = c1 + T(2-1) + T(2-2) = c1 + T(1) + T(0) = c1 + c0 + c0 = c1 + 2c0
\]

\[
T(3) =
\]

\[
T(4) =
\]

\[
T(5) =
\]

\[
T(n) =
\]

What is big-O of \(T(n)\)?