**Name:** **Worksheet 12: Advanced Functions** **CSCI 128**

1. Consider the following code:

```
1  def recursiveFunction(n):
2      if(n == 0):
3          print(1)
4          return 1
5      else:
6          x = recursiveFunction(abs(n) - 1) * n
7          print(x)
8          return x
9
10 if __name__ == "__main__":
11     num = recursiveFunction(4)
12     print("Num:", num)
```

(a) Why is the if statement on line 2 necessary for the code to function correctly?

It's the base case for the recursive function.

(b) Once the code is run, what will be printed to the terminal?

```
1
1
2
6
Num: 24
```

(c) This recursive function models a simple mathematical function, what is it?

A factorial

2. The fibonacci sequence is a sequence defined by the fact that the nth number is the sum of the two numbers in the sequence before it, and the 0th and 1st numbers are 0 and 1 respectively. So:
0th = 0,
1st = 1,
2nd = 0th + 1st = 0 + 1 = 1,
3rd = 1st + 2nd = 1 + 1 = 2,
4th = 2nd + 3rd = 1 + 2 = 3,
5th = 3rd + 4th = 2 + 3 = 5,
etc

The recursive function should calculate the value of the nth number in the fibonacci sequence:

```python
def fibonacci(n):
    #base case
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        # Missing: Your line of code here

if __name__ == "__main__":
    num = int(input())
    print(fibonacci(num))
```

What should the missing line of code be?

return fibonacci(n-1)+ fibonacci(n-2)

3. A company has a database where it keeps track of all its clients. Each client has a numerical client ID and a name, as well as other important information about the client. The database is organized in order by client ID. Imagine someone trying to search this database for a particular client.

   (a) If they only have the client ID, what search algorithm should they use, of the two below?
   [✓ A. binary search] B. linear search

   (b) If they only have the client name, what search algorithm should they use, of the two below?
   A. binary search [✓ B. linear search]

   (c) Explain the reasoning for your answers above

   Binary Search is More Effective for ordered data.
   Linear is good at data with no specific order.

4. Consider Binary search, a faster alternative to linear search, in more detail. Based on your understanding of the algorithm so far, write pseudocode for a binary search algorithm that returns if an element is present in a list or not. Binary search works well either iteratively or recursively, but we recommend attempting a recursive approach here.

5. Consider the following problem:

   You want to construct a long plank using smaller wooden pieces. There are three kinds of pieces of lengths 1, 2, and 3 ft respectively, which you have an unlimited number of. You can glue together several of the smaller pieces to create a longer plank.

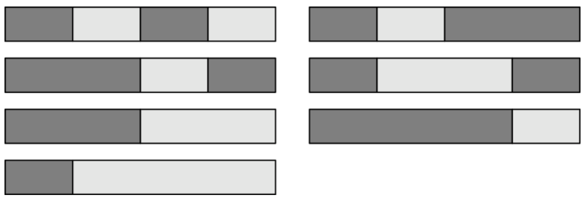

Figure 1: There are 7 ways to make a 4 ft plank.

   If the plank should have a length of $n$ feet, in how many ways can you glue pieces together to create a plank of length $n$?

   (a) Write a recursive solution for this problem (pseudocode). Hint: if you have a plank of length 4, what could you have added to some smaller plank length(s) to create a plank of length 4? See the figure above.

(b) Are there any similarities between this solution and the fibonacci solution above?

(c) Implement your recursive solution in Python as part of the studio problems for this week.