**Design 5 Template**

**Part 1: Recursion**

**Understanding the problem:**

We are given a staircase of *n* number of steps. Using 3 predefined ways of climbing the stairs (taking one small step, taking one medium step – skipping 1 small step, and taking one large step – skipping 2 small steps) we are tasked with recursively determining the number of ways that we can get to exactly the top of the staircase of *n* steps.

**To make sure that you understand the problem correctly, answer the following guiding questions:**

1. **(6 pts) What would ways\_to\_top (1) return? ways\_to\_top (2)? ways\_to\_top (6)? List them out, like the examples provided in the assignment.**

ways\_to\_top(1) would return a statement saying that there is one way to get to the top. ways\_to\_top(2) would return a statement saying that there are 2 ways to get to the top. ways\_to\_top(6) would return a statement saying that there are 24 ways to get to the top.

**(1 pt) [List assumptions that you are planning to make for this program]**

I am assuming that this program will only process integer values, and won’t be processing floats or other datatypes.

**Program Design:**

To help you break the problem down into smaller subtasks, answer the following guiding questions:

1. **(1 pt) What is the general pattern of this recursion problem? i.e., how can you break the problem down into smaller versions of the same problem?**

The general pattern of this recursion problem is to divide the total number of steps into subgroups that you can better work with. You can divide it into subgroups that are easily divisible by 1, 2, or 3, for example.

1. **(2 pts) For a general function call of ways\_to\_top(N), how many recursive calls should it have? And what are they?**

For a general function call of ways\_to\_top(N), the function should generally have 3 recursive calls, for calls where N is greater than 2. This grows exponentially with larger numbers. The calls are ways\_to\_top(N-1).

1. **(1 pt) What is / are the base case(s) of ways\_to\_top()?**

The base cases of ways\_to\_top(N) are any case where N is less than or equal to one.

1. **(1 pt) How many helper functions are you going to implement? What is the purpose of each of them?**

I will need to implement two helper functions; a helper function to calculate the results, and a helper function to store the results to display them.

**(4 pts) [Create pseudocode (or flowchart) for ways\_to\_top (N) and each of your helper function]**

ways\_to\_top:

**Program Testing:**

To help you consider the possible test cases, answer the following guiding questions:

1. (1 pt) What is an example of good input?

**Good input is an integer value.**

1. (1 pt) Are there any inputs that would cause your program to crash?

**Non-integer input – specifically strings and chars.**

1. (1 pt) What if the inputs are too large for your recursive problem to run?

(4 pts) [Create a testing table that has representative good, bad, and edge cases, and their expected outputs]

**Part 2: Linked List Implementation**

**Understanding the problem:**

(1 pt) [Rephrase the problem statement]

To make sure that you understand the problem correctly, answer the following guiding questions:

1. (2 pts) What would each Node object store? What does the next pointer point to?
2. (2 pts) In Linked\_List class, what’s the purpose of the head pointer? Why is it a pointer instead of a Node object?

(2 pts) [List assumptions that you made]

**Program Design:**

To help you break the problem down into smaller subtasks, answer the following guiding questions:

1. (2 pts) How would you iterate through the entire linked list? How could you tell if it reaches the end of the list?
2. (1 pt) How would you swap two nodes in a given linked list? Write down the steps in pseudocode.
3. (1 pt) Can any of the listed functions serve as a helper function to others? If so, how?
4. (1 pt) How will you implement the merge sort algorithm to work on a linked list in sort\_ascending()?
5. (1 pt) What algorithm are you going to use for sort\_descending()? How will you implement it to work on a linked list?
6. (1 pt) What other private member variables are you planning to add? What is the purpose of each?
7. (1 pt) How many helper functions are you going to implement? What is the purpose of each of them?

(4 pts) [Create pseudocode (or flowchart) for every function listed in the document, as well as your helper function(s)]

**Program Testing:**

To help you consider the possible test cases, answer the following guiding questions:

1. (1 pt) What is an example of good input?
2. (1 pt) Are there any inputs that would cause your program to crash?
3. (1 pt) What is your understanding of the providing test cases?

(4 pts) [Create a testing table that has representative good, bad, and edge cases, and their expected outputs. (You may use the provided test cases as reference, and you are encouraged to create your own test cases)]