Name: *John Hellrung & Ivan Santos*

This reflection is to be completed individually, though consultations with TAs and classmates are encouraged as long as they are appropriately acknowledged.

Included with the assignment are implementations of the BinaryNumber class in both Python and C++ in which binary numbers are represented using linked lists bit nodes for each bit of the number. Each node in that list is essentially a bit of that number, and adding one to that number can result in a "ripple effect" of carries, which can increase the length of the list.

1. The binary representation had the head of the linked list as the least significant bit (i.e. the number 13 in base 10 is represented as 1🡪0🡪1🡪1.) Identify one significant advantage this representation has with regards to how this number can be incremented.

|  |
| --- |
| With the numbers being flipped, the python interpreter is able to iterate through the list as if it was being read backwards. |

1. Identify two other possible representations that one might use to represent a binary number. For each, determine whether you think that adding would be easier and why.

|  |
| --- |
| One representation would be calling a direct number and incrementing it then using the binary method and changing the number to binary. This would be easiest because it works with the number directly.  Second representation would be working with it through a linked list; where the number is determined by the binary state. The state will change and the number will be dependent on the binary value. This is a more complex method of incrementing, but it allows one to understand the process. |

1. Notice that in the convert\_decimal\_to\_binary() function, it refers to the leastSignificantBit instance variable only in the first half of the function, and the rest referred to bitRef. Explain why the function still works even though it does not refer to leastSignificantBit all the time.

|  |
| --- |
| The reason why bitRef still works is because earlier in the code leastSignificantBit was equal to bitRef thus meaning that when bitRef is called later in the code it is being used as a deep copy of leastSignificantBit method. |

1. Explain the significance of the arrow (🡪) notation for the nodes of the linked list.

|  |
| --- |
| The arrow identify the reference to the one node of memory to the next node of memory. |

1. Compute the Big-O analysis for the increment()method as you implemented it, explaining the computation.

|  |
| --- |
| The computation for your code is O(1) because we increment the linked list by one. |

1. Describe the primary conceptual challenges do you feel that you encountered in trying to complete this assignment and what might have made the assignment easier for you.

|  |
| --- |
| The primary conceptual challenge was to understand how to use the bit.py file. The bit.py file had everything we needed to use; however, the instructions on how to use them were fuzzy. The way that this assignment would have been easier would be by including comments on what the methods do and how to use them. |