***Marima Andrew Mambondiumwe***

***CSC 236 A10: Binary Numbers and Linked Lists reflection***

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-None.) Identify one significant advantage this representation has with regards to how this number can be incremented.**

In binary, bits are added starting from the least significant position therefore since the head is our least significant bit, we do binary addition starting from the head.

1. I**dentify 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.**

A binary tree can be used to represent a binary number. The branches of the tree would be wither 0 or 1 and The branches would be none. The number of bits we add would increase in direct proportion as we go further down in the binary tree. Adding would be easier because we would have to traverse through the tree only once.

A stack uses a last in- first out system and as such they can be used to represent a binary number, the bottom of the stack would have the most significant bit and the top of the stack will have the least significant bit. So, we can extract the top of the stack to add a number to it and store it in a different stack until we iterate through all the numbers in the stack. From there we then go to a different stack and reverse the order of the stack by putting in the old stack. Using stacks would make adding easier than when using linked lists.

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.**

bitRef has a reference and a value, therefore, it is an object. When the reference to the least significant bit has been stored in bitRef, the least significant bit will no longer be of use anymore.

1. **Explain the significance of the arrow (->) notation for the nodes of the linked list.**

The arrow notation shows the reference to the next node for the current node.

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

O(n). n is the input and to traverse through all the bits will take n steps. One additional step will be added if a new bit is created, but if n is very large Big O complexity is O(n).

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

Trying to tie in the knowledge I got from the section on Big-notation was at first a bit confusing but after revisiting that particular section, things became clearer. I think I just need to keep doing more work on linked lists and the Big-O notation.