Assignment 5: Design **John Waczak**
CS 162 Date: June 12, 2019

Understanding the problem

In your own words, explain what YOU think the problem is asking you to do. Document your uncertainties about the problem and anything else that you feel was unclear or vague

This assignment asks us to implement our own singly-linked list class for ints. The list will consists of a group of nodes, each of which contains data (an integer) and a pointer to the next node. The first node is called *head* and the final node is called the tail. This final tail node has the next pointer set to NULL.

Member functions for the list class include an ascending merge sort implementation and a descending sort (can use recursive selection sort for extra credit). We must also be able to insert values at the head, tail, or a specified index. Finally, we must also write a function to determine how many items in the list are prime numbers.

Once everything is working, we are to showcase the functions in an driver file.

Devising a plan/design

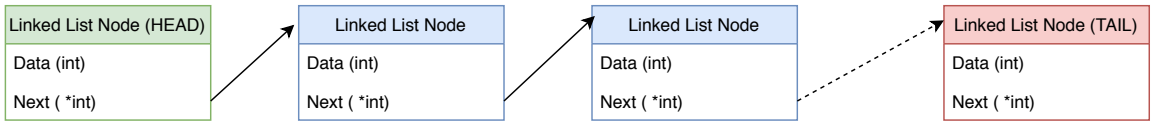
Provide an algorithm/pseudo code to help solve the problem. In addition, draw pictures/flow charts to help you devise your plan, as well as any other design decisions you make, such as how to manage your time, how to decompose the problem, where to start first, etc.

The following code snippet illustrates the class structure:

```
class Linked_list_node:
  public
    val
    Linked_list_node * next

class Linked_list:
  private
    length
    Linked_list_node * first
  public
    print()
    clear()
    push_front(value)
    push_back(value)
    insert(value, index)
    sort_descending()
    sort_ascending()
    get_num_primes()
```

The following figure illustrates how the linked list nodes are connected.



For the sorting algorithms, I will use my work from Lab9. For detecting primes, we can do something like

```
function is_prime(n):
  if(n<=1):
    return false

  for (i = 1 to n-1):
    if (n%i==0):
      return false
  return true
```

Although this is highly inefficient.

Looking back / testing

This includes any checking/self-reflection you did while solving the problem, which includes using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances, using sources of information to make sense of the results, etc. However, you need to think about the input prior to implementation!!!

Input	**Expected**
is_prime(4294963943)	true
is_prime(1)	false
is_prime(any negative number)	false
push_front(12)	return length of list + 1
push_back(12)	return length of list + 1
insert(12, any_index)	return length of list + 1