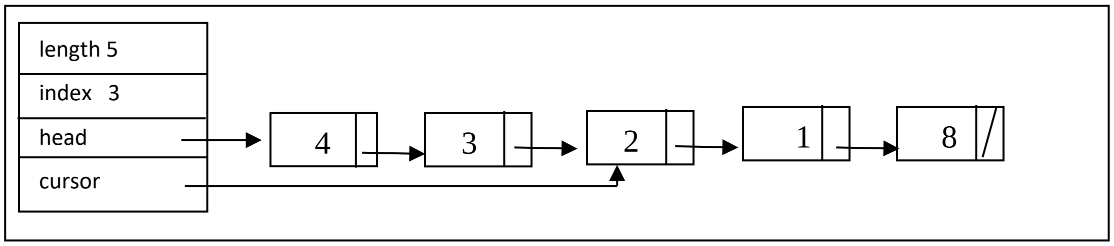
CMPS 1063 – Data Structures & ADTs **Fall 2024**
Program Assignment 5 100 points **DUE: Tuesday, Nov. 26th, 2024**

Purpose: To learn how to use dynamic memory allocation and linked list.

Problem: In this assignment, you are to process super cardinals, that is *big numbers*. You will test methods of the **Supercard** class using a test driver, and then you will find the 10's complement of a number, sum two super cardinal numbers, and print them out. NOTE: This is basically still and UnsortedList, I just changed the name.

Method: Very large integer numbers cannot be stored in variables of type int or even long int. Certainly one cannot perform arithmetic, e.g. addition, multiplication, etc., on very large numbers. For example, one cannot compute the factorial of 500 or 5^{100}. For this assignment, you will implement a class and methods to access the digits in a super cardinal (a linked list of digits). Suppose the supercard S is 81234. In your representation of S, the digits of S will be stored in a linked list with the least significant digit first; that is, in reverse order. Supercard 81234 is pictured as follows:



Note the digits are stored backwards to make it easier to perform addition.
In addition to representing the digits of S, the representation will also keep track of **four** other pieces of data.

- **head**, a pointer to the front of the list (the least significant digit of the number; remember the digits are stored in reverse in the list)
- **cursor**, a pointer to some digit in the list
- **length**, which represents the number of digits in S.
- **currentIndex**, which is described as follows:
  Let S = $d_n d_{n-1}..d_2 d_1$ where d_i is the ith digit of S. For example if S = 81234, then d_1 = 4, d_2 =3, d_3= 2, d_4 =1, and d_5 = 8. In the conceptual view of the linked list S could be represented as
  4 3 ↓ 2 1 8 with the cursor currently indicating 2, which is d_3. In this case the current index is 3. Current index is used to improve the efficiency of the implementation of the getDigit and setDigit methods in the Supercard class. The need for this information arises from the fact that getDigit and setDigit are conceptually "random access" operations for Supercard, but the implementation of supercard, a linked list, is not a random access structure like an array.

The type declaration for the representation of Supercard shown above are:

```
class Supercard
{
    private:
       int length;
       int currentIndex;
       Node * head;
       Node * cursor;
}
```

where Node is declared as a class inside Supercard as:

```
class Node
{  friend Supercard;        //this allows Supercard to access the private data
   int item;                //this is an integer digit
   Node* next;

   Node (int item1, Node* next1 = NULL) {item = itemValue; next = p;}
};
```

You will find a zipped project on D2L for this assignment. Unzip it. In the project, there are 3 files: Supercard.h, Supercard.cpp, and TestSupercard.cpp (the main file). Print out the Supercard.h file and study it – it specifies a list of methods for Supercard. There are also comments and some code in the other files. You are to complete the implementation of Supercard and complete the implementation of three functions in main: printSupercard (recursive print), complement and addSupercards.

Keep the following ideas in mind when your implement the Supercard methods:

- The digits of Supercard are stored in a linked list, but you can only go forward in the list. Reset the cursor and current index and then move/update them to get to the desired position.
- A Supercard ***always*** has at least one node in it – it may be just 0.
- Comment out parts of the main program until you get the constructor, getLength and getDigits and setDigits working.
- Then get printSuperCard to work.
- Get the methods swap and the destructor to work.

Finally, you must implement two other NON-MEMBER functions: tensComplement and addSupercards.

The 10's complement is easy, replace each digit with 10 – digit, so the 10's complement of 84321 is 26789.

Adding two Supercards is done by adding corresponding digits starting with the least significant digits with proper carry from previous digits. Think about how you added big numbers in the 3rd grade using the usual paper-and-pencil method. The result of the addition is stored in another Supercard. The function must accept as parameters two Supercards and it returns a Supercard with the digits for the sum. Upon returning from the function, the program should print the result.

Input: None, you are running a test driver. The main program will initialize the Supercards.

Output: Output is to a file.

Extra Credit (10 points)
Write the non-member function in the main program addSupercards using recursion. (5 points)

Turn in: Printouts of all .h and .cpp files (3 files)
Printout of the output file – turn this in EVEN if your output is incorrect!