CSCE 1101 Fall 2022: Fundamentals of Computing II
Assignment #5

Dr. Amr Goneid ***Date: Thu Dec 1, Due: Thu Dec 8, 2022***

In this assignment, you need to be familiar with the subjects of Sorting Algorithms and Simple Liked Lists

---

Part (A): Sorting Algorithms

Problem (1): Insertion Sort (20 points)

The following code segment performs Insertion Sort on an array a[ ] of size (n) with the elements located in a[0], a[2], …., a[n-1]:

```
for (i =1; i < n; i++)
        {   v = a[i]; j = i;
            while(j > 0 && a[j-1] > v)  { a[j] = a[j-1]; j--; }
            a[j] = v;
        }
```

- Sort the character list a ={S, O, R, T, I, N, G} in alphabetical order using Insertion Sort.
- Find the exact number of array element comparisons done by this algorithm in the for the above list.
- Find the exact number of array element moves done by the algorithm for the above list.

Problem (2): MergeSort (20 points)

The array of integers: (4 , 7 , 6 , 8 , 3 , 2 , 1 , 5) is input to the Mergesort algorithm.

- Trace the algorithm to sort the array. Use a table to show the tracing steps, e.g.

4	7	6	8	3	2	1	5
..							
..							
..							

- What is the height of the tree resulting from the Divide & Conquer method used in the algorithm?
- Explain why MergeSort has a complexity of ***O(n log n)***

Part (B): Simple Linked Lists

In the following problems, assume that you have a Simple Linked List (SLL) class named ***List*** with the following definition:

```
template <class keyType, class dataType>
class List
{
        public:
        // Member Functions
        List();          // Create an empty List
        ~List();         // Class Destructor
        // Functions Prototype Definitions
        bool listIsEmpty() const;
        bool curIsEmpty() const;
        void toFirst();          bool atFirst() const;
        void advance();          void toEnd();
        bool atEnd() const;   int  listSize() const;
        void updateData (const dataType & );
        void retrieve (keyType &, dataType &) const;
        void insertFirst (const keyType &, const dataType & );
        void insertAfter (const keyType &, const dataType & );
        void insertBefore (const keyType &, const dataType & );
        void insertEnd (const keyType &, const dataType & );
        void deleteNode();     void deleteFirst();
        void deleteEnd();      void makeListEmpty();
        bool search (const keyType & );
        void orderInsert(const keyType &, const dataType & );
        void traverse();
        private:
        // Node Class
          class node
          {
                public:
                        keyType key;                    // key
                        dataType data;                  // Data
                        node *next;             // pointer to next node
           }; // end of class node declaration
        typedef node * NodePointer;
        NodePointer head, cursor, prev;     // Pointers
    }; // End of class List declaration
```

Also Assume that both key and data fields are integers

Problem (3): SLL (20 points)

Implement a ***user application*** function to receive a non-negative integer **n** and return a list in which each decimal digit of the number is placed in the data field in a node by itself and the order of the digit is to be put in the key field of the node. The order of the nodes should preserve the order of the digits in the original integer from left to right.

Problem (4): SLL (20 points)

Implement a ***user application*** function to receive two lists (L1) and (L2) and append (L2) to the end of (L1), returning the result in a new list (L) without changing the original lists.

Problem (5): SLL (20 points)

Add ***a public member*** functions to the ***List*** class: ***SameAs(List &L2)*** to return true if the list is the same as another list ***L2***
