CSC 340.02 - Spring 2022- Assignment 4
Due: Tue May 3, 11:55pm
Grade: 75pt - 15%
Extra credit: 25pt: 5%

Guidelines for the assignment submissions:

What to submit?

- One zip/tar.gz file:
  - Name: `<First Name><Last Name>-Assignment-##.zip`
  - Source code should include only files that we create and edit (*.cpp and/or *.h)

How to submit?

iLearn, Assignment Submission section

1. Linked list: (50pt)

   Using the provided source files, implement the following functions for our linked list within LinkedList.h. You may use aClient.cpp to test your implementation. (50pt)

   1.1. copy constructor
   1.2. assignment operator
   1.3. operator==
   1.4. operator+
   1.5. replace function

2. (25 pt)

   Mark and Jane are very happy after having their first child. Their son loves toys, so Mark wants to buy some. There are a number of different toys lying in front of him, tagged with their prices. Mark has only a certain amount to spend, and he wants to maximize the number of toys he buys with this money. Given a list of toy prices and an amount to spend, determine the maximum number of toys he can buy. Note each toy can be purchased only once.

   - Input: Enter the dollar amount Mark can spend: 50
     Enter the number of items: 7
     Enter the toy prices: 1 12 5 111 200 1000 10
   - Output: Maximum number of items Mark can buy: 4

   NOTES: you're not allowed to use any standard library functions in your solution.

3. Indexed List: (25pt)

As presented, link-based lists must be searched sequentially. For large lists, this can result in poor performance. A common technique for improving list searching performance is to create and maintain an index to the list. An index is a set of pointers to various key places in the list.

For example, an application that searches a large list of names could improve performance by creating an index with 26 entries - one for each letter of the alphabet. A search operation for a name beginning with “K” would first search the index to determine where the “K” entries begin and “jump into” the list at that point and search linearly until the desired name was found. This can be much faster than searching the linked list from the beginning.

Using the provided source file, create an IndexedList class that holds names and an index of 26 entries as the above example. This class provides the following member functions:

3.1. insertInIndexedList: insert the provided name at front of all names with the same index

For example:

- If your current list is:
  Bobby -> Joe -> James -> Kevin -> Katherine -> Stacy -> null
- After inserting Kalvin, your list should be:
  Bobby -> Joe -> James -> ***Kalvin*** -> *Kevin* -> *Katherine* -> Stacy -> null

3.2. searchInIndexedList
3.3. deleteFromIndexedList
3.4. operator<< to output data per index.