CS-224: Object Oriented Programming & Design Methodologies

Assignment 04

Duration: 1 Weeks (Please submit it by 19 October before 1159 pm)

This assignment is a practice in Operator Overloading. You will be provided a bare minimum code for a class *mylist* implementing a linked list of integers wrapped in *struct node*. For our convenience, we'll say that if the list has N elements then these elements are placed at indices ranging from 0 to N-1. The default constructor creates an empty list. You can populate it by adding elements using the append function.

TO DO:

First, you should define proper constructor, destructor, and copy constructor for the class *mylist*.

You are to overload the following operators for the *mylist* class such that their behavior is as explained below:

If l and q are two objects of class *mylist*:

1. The = operator (assignment):

   l = q; // should copy all the values in list q to list l. The list q should not be changed.

2. The + operator:

   l = l + 23; //This should append an element at the end of the list and store the value 23 in it.

   // if q is also a list of type mylist

   l = l + q; // This should append the list q at the end of the list l

3. The – operator:
   l = l – 23; //This statement should search for all the occurrences of the value 23 in the list and remove all of them from the list.

4. The [] operator:

   The user should be able to access the elements of the linked list using the array subscript notation, i.e. [], for reading as well as writing values. E.g:

   cout <<l[3]; // this should read the value stored at index 3, i.e., the fourth element of the list

   l[3] = 23; // this statement should write the value 23 in the 4th element of the linked list

   If the indices are out of range, make then wrap around the array bounds, i.e., l[-1] should access the last element of the list and l[N] should access the first, i.e., l[0]. You'll need to handle the special case of an empty list, i.e., the best way that you think a list should behave.

5. The << operator:
   cout <<l; //This should display all the elements in the list to output screen enclosed in {} brackets separated by commas i.e. if there were there elements in the list with values 7, 8 and 10, this statement should display {7,8,10}

6. The == operator:
   l==q should return true iff the number of elements in both lists are equal and the values stored in elements at corresponding indices are equal.

7. There is an attribute *size* in class mylist. We do not strictly need it. Every time you need the length of the linked list, you can calculate it. So remove the attribute *size* from the class. Your resulting class should have only one data member *start.*

Make sure that you DO NOT create any memory leaks or dangling pointers.