Assignment 3 – Data Structures

For this assignment, you will solve problems based on what you have learned in Data Structures.

Instructions

- Review notes of the Chapter.
- There are 3 questions in this assignment.
- **Assignment submitted after due date will not be evaluated and a score of zero will be awarded for this assignment.**
- Combine all files into **one zip file**, which you submit via BB.

Due Date: Midnight, March 16, 2020.

Submitting this Assignment

- You will submit (upload) this assignment in Blackboard. Email/paper submissions will not be accepted.
- Name the zip file as "A2_2020_John_Doe" in case your name is "John Doe".

Grading Criteria

This assignment has 02 points (with weightage of 02% in your overall 100 points).

Questions:

Ques. 1 Implement a data type ***List*** that represents linked lists consisting of nodes with integers. Remember that:

- an object of type ***Node*** has two fields, a data field and a pointer field
- an object of type ***List*** has one field, (a pointer to) a Node ***head***

The type ***List*** must have the following functions:

1. **boolean isEmpty( );**

2. **int length( );**
   Returns the number of nodes in the list, which is 0 for the empty list;

3. **void print( );**
   print the content of all nodes;

4. **void addAsHead (int i);**
   creates a new node with the integer and adds it to the beginning of the list;

5. **void addAsTail (int i);**
   creates a new node with the integer and adds it to the end of the list;

6. **void addSorted (int i);**
   creates a new node with the integer and adds it behind all nodes with data less or equal to the data of the node, possibly at the end of the list;

7. **Node find (int i);**
   Returns the first node with data i;

8. **void reverse( );**
   reverses the list;

9. **int popHead( );**
   returns the value of the head of the list and removes the node; if the list is nonempty, otherwise returns NULL;

10. **void removeFirst (int i);**
    removes the first node with value i;

11. **void removeAll (int i);**
    removes all nodes with val i;

12. **void addAll (List L);**
    appends the list L to the last element of the current list, if the current list is nonempty, or lets the head of the current list point to the first element of L if the current list is empty.

Ques. 2 Implement a data type ***HTList*** that represents head-tail lists consisting of nodes with integers. Remember that:

- an object of type ***Node*** has two fields, a data field and a pointer field
- an object of type ***HTList*** has two fields, (a pointer to) a Node ***head*** and (a pointer to) a Node ***tail***

Modify the functions for the type List from the first question so that they work for ***head-tail*** lists. You may find that some methods need not be changed at all, while for others you also have to manage the ***tail*** pointer.

Ques. 3 Implement Insertion Sort for Linked Lists. Implement it as a function

void insertionSort() that turns the current list into a sorted list.