Data Structures: Homework

MTH 9815: Software Engineering for Finance

Note: Please use the C++ coding standards as specified in the following guide:

https://google.github.io/styleguide/cppguide.html

DUE DATE: THURSDAY, NOVEMBER 21, 2024 at 9pm

Please reach out to me on the forum should you have any questions. You can complete this homework in the modified groups on the forum (see post). **NOTE that these homework groups are not exactly the same as Dan's (but very similar), hence see post on homework groups!**

NOTE: All code must run on UNIX with a g++ version of 7.x or greater. You must provide detailed instructions on compilation with makefile to compile your code. CMake should also be used to set up your makefile.

Well-documented and commented code is essential to understand what your code does, and will be comprise a portion of the grading.

Use of Chat GPT is allowed as part of this homework. Please also document how you used Chat GPT if you did so as part of the documentation of the submission.

EXERCISE 1

Write a singly linked list data structure in C++. You should define the following three classes:

```
template <typename T>
class Node;

template <typename T>
class LinkedList;

template <typename T>
```

```
class ListIterator;
```

Define the following methods:

```
// Add the specified element at the end of the list
template <typename T>
void LinkedList<T>::Add(T& value);

// Add the specified element at the specified index
template <typename T>
void LinkedList<T>::Insert(T& value, int index);

// Get the element at the specified index
template <typename T>
T& LinkedList<T>::Get(int index);

// Retrieve the index of the specified element (-1 if it does not
exist in the list
template <typename T>
int LinkedList<T>::IndexOf(T& value);

// Remove the element at the specified index and return it
template <typename T>
T& LinkedList<T>::Remove(int index);

// Return an iterator on this list
template <typename T>
ListIterator<T> LinkedList<T>::Iterator();

// Return the size of the list
template <typename T>
int LinkedList<T>::Size();

// Return whether there is another element to return in this iterator
template <typename T>
bool ListIterator<T>::HasNext();

// Return the next element in this iterator
template <typename T>
T& ListIterator<T>::Next();
```

EXERCISE 2

Write a doubly linked list data structure in C++. You should define the following two classes and reuse the iterator above (inheritance recommended but not necessary for the doubly linked list and node):

```
template <typename T>
class DNode : public Node<T>;

template <typename T>
class DoublyLinkedList : public LinkedList<T>;
```

Define all methods that were defined above on the `LinkedList`.

EXERCISE 3

Write a hash table class that maps keys to values. Define the following classes:

```
template <typename K, typename V>
class Hashtable;

template <typename K>
class Hasher;

template <typename K>
class EqualityPredicate;
```

The `Hasher` and `EqualityPredicate` classes should be base classes with pure virtual functions. They can be overridden to provide concrete implementations of a hashing function and equality predicate function, respectively.