Assignment 2

Due Date: 5 May 2023

Weighting: 30%

Group or Individual: Individual

This assignment is to assess your knowledge about non-linear data structures and algorithms and your skills in applying the knowledge in the development of reusable Abstract Data Types (ADTs). This assignment also assesses your non-linear data structure-based algorithm design and analysis techniques.

In this assignment, you are given two ADT specifications in C# interfaces (*IMember.cs*, *IMovie.cs* and *IMovecollection.cs*) and a skeleton of the ADT implementations (*Movie.cs* and *MovieCollection.cs*). Your task is basically to complete the two ADT implementations. When implementing the ADTs, you need to design an efficient algorithm to solve two computational problems arising in the ADT implementations, to analyse the time efficiency of the algorithms, to test the ADT implementations, and to write a technical report.

Movie is an ADT that models a movie DVD in the library. A movie has a title, genre, classification, duration, the total number of copies of the movie DVDs in the library, and the number of available DVDs of the movie currently in the library.

MoiveCollection is another ADT, which is used to store and manipulate all the movies in the library. The basic data structure that is used in the *MovieCollectio*n implementation is a Binary Search Tree (BST). A node of the BST is modelled by an inner class, namely *BTreeNode*, in *MovieCollection.cs*. A node has three fields, left child reference (*lchild*), right child reference (*rchild*) as well as a *Movie* object. Each node in the BST is an instance of *BTreeNode* class.

It is assumed in this assignment that the movie titles are unique, the full names of the members are also unique. In addition, we do not need to distinguish the multiple DVDs of the same movie in this assignment.

Your detailed jobs in this assignment are:

1. Use the pseudocode notation introduced in this unit (Lecture 1) to design an efficient algorithm for finding the total number of DVDs in the library, empirically analyse the time efficiency of the algorithm, and use this algorithm to implement the method "*public int NoDVDs()*" in *MovieCollection.cs.*

2. Complete all the other incomplete methods in the ADT implementation skeletons (*Movie.cs* and *MovieCollection.cs*).
3. Design a test plan for each of the methods that you implemented.
4. Use the test plan to comprehensively test each of the methods that you implemented.

5. Write a technical report that includes, but is not limited to, the following:
    - Cover page
    - Table of contents
    - Algorithm design and analysis
    - Test plan, test data and test results
    - References

1. Assignment Requirements

- The programming language used in this assignment must be C#.
- You must not use any third-party C# class libraries. Types and methods defined in the *System.Collections*, *System.Collections.Generic*, and *System.Linq* namespaces are specifically prohibited.
- You may use any code in the CAB301 lectures and workshops.
- You must not make any change to any of the given C# interfaces.
- You must not make any change to the class fields in the skeleton ADT implementations.
- The pre-conditions and post-conditions for each of the methods that you need to implement in this assignment are given in the ADT specifications. The invariants of each ADT are also given in the ADTs specifications. A precondition is a condition, or a predicate, that must be true before a method runs for it to work. In other words, the method tells clients, “this is what I expect from you”. So, the method we are calling is expecting something to be in place before or at the point of the method being called. The operation is not guaranteed to perform as it should unless the precondition has been met. A postcondition is a condition, or a predicate, that can be guaranteed after a method is finished. In other words, the method tells clients, “this is what I promise to do for you”. If the operation is correct and the precondition(s) met, then the postcondition is guaranteed to be true. An ADT invariant is something that is always true and won’t change. The method tells clients, “if this was true before you called me, I promise it’ll still be true when I’m done”. An invariant is a combined precondition and postcondition. It has to be valid before and after a call to a method.

2. Submissions

- Your submission should be a single zip file named by *your-student-number*.zip. The submitted archive must be standard .zip format. Uploads in other formats such as .7z, .rar, .gz, etc, will not be accepted.
- Your submission must be submitted from CAB301 Canvas website. Email submissions are not accepted.
- You can submit your assignment once before the deadline.