Development of a Software Application for a Community Library to Manage Its Movie DVDs

Phase 1

Due Date: 27 April 2022

Weighting: 25%

Group or Individual: Individual

1. Introduction

In this project you develop a software application to solve a real-world problem. In the development of this software application, you need to use some data structures and algorithms that are covered in this unit to store, manage, and manipulate the data in the software application. In this software application development, you also need to design algorithms to solve some computational problems in the software application and to analyse the time efficiency of your algorithms.

This project will be completed in three phases. In this phase (Phase 1), you need to develop some reusable Abstract Data Types (ADTs), which will be used as components to build the software application in Phase 3.

By now, the requirement analysis of the software application has been completed, and the ADTs that need to be used to build the software application have been identified. For those who do not know the concept of ADT, an ADT is defined by its behaviours from the point of view of a user, of the data, specially in terms of possible values, possible operations on data of this type, and the behaviours of the operations.

In C#, an ADT may be defined as a C# interface, which is nothing but a collection of methods and property declarations. The C# interface must be implemented before the ADT can be used in a software application.

In this assessment, you are provided two ADT definitions in C# interfaces, *IMember.cs* and *IMemberCollection.cs*, and the skeletons of the corresponding ADTs' implementations, *Member.cs* and *MemberCollection.cs*. These C# interfaces and incomplete C# classes can be downloaded from *Assessment Task 1* under *Assessment Tasks* in the CAB01 Blackboard. Your tasks in this phase are to complete the ADTs' implementation. Below are the details:

2. Detailed Tasks

- Design an efficient algorithm to check if a phone number is valid. The phone is stored in a string, which is considered as an array of chars. A phone number is valid if it contains exactly 10 digits, and the first digit is 0. Use the pseudocode notations introduced in Topic 1 Lectures to formally describe your algorithm, then theoretically analyse the time efficiency of your algorithm, and then apply your algorithm to the implementation of the static *IsValidContactNumber* method in *IMember.cs* (a static method specified in a C# interface must be implemented in the interface).

- Design an efficient algorithm to check if a pin is valid. The pin is stored in a string, which is considered as an array of chars. A pin is valid if it has a minimal of 4 digits and a maximal of 6 digits. Use the pseudocode notations introduced in Topic 1 Lectures to formally describe your algorithm, then theoretically analyse the time efficiency of your algorithm, and then apply your algorithm to the implementation of the static *IsValidPin* method in *IMember.cs* (a static method specified in a C# interface must be implemented in the interface).
- Design an efficient algorithm to add a *Member* object into a sorted array of *Member* objects such that after the addition those *Member* objects are still sorted in the dictionary order by the members' last names, and if there are two members who have the same last name, then they are sorted by their first names. Use the pseudocode notations introduced in Topic 1 Lectures to formally describe your algorithm, then theoretically analyse the time efficiency of this algorithm, and then apply your algorithm to the implementation of the *Add* method in *MemberCollection.cs*.
- Design an efficient algorithm to delete a *Member* object from a sorted array of *Member* objects such that after the deletion those *Member* objects are still sorted in the dictionary order by the members' last names, and if there are two members who have the same last name, then they are sorted by their first names. Use the pseudocode introduced in Topic 1 to formally describe your algorithm, then theoretically analyse the time efficiency of this algorithm, and then apply your algorithm to the implementation of the *Delete* method in *MemberCollection.cs*.
- Design a binary search algorithm to search for a *Member* object in an array of *Member* objects that are sorted in the dictionary order by the members' last names, and if there are two members who have the same last name, then they are sorted by their first names. Use the pseudocode introduced in Topic 1 to formally describe your algorithm, then theoretically analyse the time efficiency of this algorithm, and then apply your algorithm to the implementation of the *Search* method in *MemberCollection.cs*.
- Comprehensively test the above five implemented methods using a variety of test data, including normal test values and boundary values, to make sure they meet the functional requirements, to check if the pre-condition(s) and post-condition(s) are satisfied. The pre-condition(s) and post-condition(s) of the methods can be found in the C# interfaces.

3. Assignment Requirements

- The programming language used in this assignment must be *C#*.
- Do not make any change to *IMember.cs* or *IMemberCollection.cs* except for implementing the *IsValidContactNumber* and *IsValidPin* methods.
- Do not make any change to *Member.cs* or *MemberCollection.cs* except or implementing the *Add*, *Delete* and *Search* methods.
- Do not use any built-in search, insert, delete method or the like from any C# class when implementing those five methods.
- Do not add any namespace to *Member.cs* or *MemberCollection.cs*.

4. Submissions

- Your submission should be a single zip file named by *your-student-number*.zip, which compresses *IMember.cs*, *IMemberCollection.cs Member.cs*, *MemberCollection.cs,* and a PDF document in which you present your algorithms and show the details of your theoretically algorithm analyses.
- Do not include any other file into the zip file. If you have an approved extension for this assessment, please also submit a copy of the approval as a separate PDF file.
- Your submission must be submitted via the Blackboard. Email submissions are not accepted as any email containing a C# program, including attachment, may be blocked by QUT email server.
- You may resubmit your assignment as many times as you wish before the deadline. If you submit your assignment multiple times, we will only mark the last submission before the deadline.