Assignment 1

Due Date: 6 April 2023

Weighting: 30%

Group or Individual: Individual

This assignment is to assess your knowledge about 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 linear data structure-based algorithm design and analysis techniques.

In this assignment, you are given three ADT specifications in C# interfaces and an ADT implementation skeleton for each of the ATDs. Your tasks in this assignment are basically to complete the implementations of the three ADTs. When implementing the ADTs, you need to design an efficient algorithm to solve some 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.

One ADT is *Job*, which is used to model a computing job in the operating system of a computer. A job has a job ID, which is integer number between 1 and 999, received time, expected execution time, and a priority, which is labelled by values ranging from 1 to 9 where 1 is the lowest priority and 9 is the highest priority.

JobCollection is another ADT, which is used to store and manipulate a collection of computing jobs. We may add a computing job into the collection, remove a computing job from the collection, check if a computing job is in the collection, or find a computing job in the collection.

A third ADT that you need to implement is *Scheduler*. *Scheduler* hosts a list of computing jobs and manage the execution of the computing jobs. Here are three commonly used strategies that are used by the scheduler to manage the execution of the computing jobs:

1. First-Come, First-Served (FCFS): Computing jobs in the job list are executed in the order they are received time, with no consideration for priority or execution time. If there are multiple computing jobs that have the same arrival time, they are executed in any order.
2. Shortest Job First (SJF): Computing jobs are executed in order of their expected execution time. If there are multiple computing jobs that have the same expected time, they can be executed in a random order. This helps to reduce the average waiting time for jobs and improves system throughput.
3. Priority: Computing jobs are executed in the order of priority. The computing job with the highest priority is executed first and the computing job with the lowest priority is executed last.

In this assessment, you are provided the specifications of the above three ADTs in C#, *IJob.cs*, *IJobCollection.cs* and *IScheduler.cs*, and a skeleton implementation (an incomplete implementation) for each of the three ADTs, *Job.cs*, *JobCollection.cs and Scheduler.cs*. Your detailed jobs in this assignment are:

1. Use the pseudocode notation introduced in this unit (Lecture 1) to describe the algorithms that need to be used when implementing the three commonly used scheduling strategies.
2. Theoretically analyse the time efficiency of the three algorithms.
3. Complete the three ADTs.
4. Design a test plan for each of the methods that you implemented.
5. Use the test plan to comprehensively test each of the ADTs.
6. 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.
- The C# interfaces provided to you in this assignment must not be modified.
- The basic data structure used in *JobCollection* must be an *Array* of *Job*. But you must not use any method in the Array class.
- The implementations of the three scheduling methods in *Scheduler* must use the corresponding algorithms that you designed in this assignment.
- 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.