National University of Singapore
School of Continuing and Lifelong Education
TIC1002: Introduction to Computing and Programming II
Semester II, 2020/2021

Tutorial 4
Additional Applications and Midterm

Note$_1$: Since Week 6's Saturday (20th Feb) will be used for PE1, this tutorial set will only be discussed in week 7's Saturday (6th March)
Note$_2$: We will go through the midterm questions as well in this tutorial. So, the number of tutorial questions is lesser than usual.

1. [Generalized Linear Searching] Although linear searching is not very efficient, it can be easily generalized. Given a sample implementation of the linear searching in `LSearch.cpp` for unsorted array, try the following:
   a. Change the code such that we look for the **first occurrence of a** number that is divisible by 7 (i.e. looks for multiple of 7, e.g. 7, 14, 21, ….)
   b. Change the code such that we can look for the **first prime number** in the list.
   c. From (a) and (b), derive a general "template" for such functions (i.e. function that looks through a list to find item that satisfy certain criteria).

2. [Counting Sort for Real] In the lecture, we discussed **counting sort** using an integer array (see the attached `CSort.cpp` for a sample implementation). Let us make it a bit more realistic by using an array of **student result** structures. The student result is captured as a C Structure:

```
struct SResult {
   string studentID;     //student id is a string with "A1234" format
   int score;          //score received by students
};
```

   Modify the Counting Sort function discussed in the lecture, so that it now handles an array of `SResult` structures. Print out the Student ID and Score arranged by Score in ascending order.

3. [Stable Counting Sort] The counting sort as discussed in lecture is not stable, why? Can you show that it is unstable using the solution of question above? Using the solution of question 2, modify the counting sort to make it stable.