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

Tutorial 3
Complexity and Sorting

1. [Not very complex complexity] Give the big-O for following code fragments

a.	`for (int i = 0; i < n; i++)`
	`  for (int j = 0; j < n; j++)`
	`  <2 operations>`

b.	`for (int i = 0; i < n; i++)`
	`  for (int j = 0; j < 2; j++)`
	`  <5 operations>`

c.	`for (int i = 0; i < n; i++)`
	`  for (int j = n - 1; j >= i; j--)`
	`  <4 operations>`

d.	`i = 1;`
	`j = 0;`
	`while (j < n ){`
	`  if (i % n == 0){`
	`  j++;`
	`  <1 operation>`
	`  }`
	`  i++;`
	`}`

2. [Selection Sort] Trace the working of selection sort on the following array. You can use the given table to show the changes after each outer-loop iteration. Indicate clearly the largest item, the location of the largest item for each iteration.

56	12	34	19	18	79	25	31

3. [Bubble Sort Version 3.0] Let us see how bubble sort can be further improved.
    a. [What's the issue?] Try sorting an array like {2, 3, 4, 5, 1}. How many outer-loop iteration do we need? Identify the issue with the standard bubble sort algorithm.

    b. [Solve the issue] Solve the issue posed by (a). Hint: It is like bubble sort with a twist….

    c. [Analyzing the change] Did we improve the big-O of bubble sort?

4. [Sorting is general] For simplicity, sorting is almost always taught using an integer array. However, it should be clear that the sorting algorithms can be easily generalized. Let us take the **insertion sort** code as a case study in this question.

    a. [What to change?] Identify all necessary changes for the insertion sort code if we need to sort a different type of array (e.g. an array of student records / double values / strings etc). Whenever possible, focus more on the higher level requirement (**"what kind of operation is needed?**") rather than low level details ("**how do I write this in C++?**")

    b. [Actual change] Using your findings in (a), change the ***insertion sort*** to work on an array of **fraction** structure as defined below:

```
struct Fraction {
    int num, den;
};
```

    Note: You should avoid converting the fractions into a floating point values for comparison.

    Note$_2$: Use the provided ***Q4-Template.cpp*** to actually code it out!