1 Introduction and Objective

The purpose of this first tutorial is to recap the first two weeks of CS2040C: Introduction, basic C++, basic analysis of algorithm, and the early part of the $O(N^2)$ sorting algorithms.

As this is the first tutorial session, we will do a quick ice breaking at the start of the session. It is important to befriend your tutorial TA (tutor) as he is about the same age as as you (or younger :O) but he already has the knowledge expertise on the subject matter.

To get the most out of the tutorial sessions, please try out all the questions in the tutorial and give some answer even if you encounter difficulties in answering some of them. Before, during, or after the tutorial session, don’t hesitate to clear up all doubts and questions you might have, with the tutor.

Btw, the tutorial participation marks to be given by the tutor at the end of the semester are simply as follows:

- 0% if you only attend $\leq 5$ out of 11 tutorial sessions (no known public holiday affecting CS2040C Monday tutorial slots in S1 AY2017/2018),
- 1% for at most the bottom three most-passive students (assuming these students attend $> 5$ tutorial sessions),
- 3% for at least the top three most-active students in each tutorial group, and
- 2% for the rest.
2 Tutorial 01 Questions

C++ OOP (basic)

Q1). You are given a simple C++ program that will be revisited later in Week04 (List ADT):

```cpp
#include <iostream> // please copy paste this code, test compile, and run it yourself using namespace std;

class ListArray {
private:
    int N;
    int A[10]; // question 1a
public:
    ListArray() : N(0) {} // question 1b
    int get(int i) {
        return A[i]; // question 1c
    }
    int search(int v) {
        for (int i = 0; i < N; i++)
            if (A[i] == v)
                return i;
        return -1;
    }
    void insert(int i, int v) {
        if ((N == 10) || (i < 0) || (i > N)) return; // question 1d
        for (int j = i; j <= N-1; j++) // question 1e
        A[i] = v;
        N++;
    }
    void remove(int i) {
        for (int j = i; j < N-1; j++) // question 1f
            A[j] = A[j+1];
        N--;
    }
    void printList() {
        for (int i = 0; i < N; i++) {
            if (i) cout << " ";
            cout << A[i];
        }
        cout << endl;
    }
};
```
void sortList() { // sort array A, question 1g

}

};

int main() {
    ListArray* LA = new ListArray(); // question 1h
    LA->insert(0, 5);
    LA->insert(0, 1);
    LA->insert(0, 4);
    LA->insert(0, 7);
    LA->insert(0, 2); // we should have A = {2, 7, 4, 1, 5} by now
    cout << LA->get(3) << endl; // 1, A[3] = 1
    cout << LA->search(6) << endl; // not found, -1
    LA->remove(1); // we should have A = {2, 4, 1, 5} by now
    cout << LA->search(4) << endl; // 1, A[1] = 4 now
    cout << LA->search(7) << endl; // not found, -1
    LA->printList(); // unsorted
    LA->sortList(); // we should have A = {1, 2, 4, 5} by now
    LA->printList(); // sorted
    return 0;
}

Now answer the following sub-questions (please refer to the comments in the C++ code above):

(a) Anything wrong with this line?

(b) What this line means?

(c) Any potential issue with this line? (also see question 1d below)

(d) What this line means?

(e) Any potential issue with this line?

(f) Any potential issue with this line?

(g) Implement this routine using any sorting algorithm that you know!

(h) Can we just write ListArray LA; in this line?
Analysis/Order of Growth

Q2). What is the bound of the following function? \( F(n) = \log(2^n) + \sqrt{n} + 10000000 \)

1. \( O(n) \)
2. \( O(n \log n) \)
3. \( O(n^2) \)
4. \( O(1) \)
5. \( O(2^n) \)

Q3.a). What is the bound of the following function? \( F(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{4}n + \ldots + 1 \)

1. \( O(2^n) \)
2. \( O(n^2) \)
3. \( O(n \log n) \)
4. \( O(n) \)
5. \( O(\log^2 n) \)
6. \( O(\log n) \)

Q3.b). What about \( G(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{4}n + \ldots + 1 \)

Sorting (part 1)

Q4). At [https://visualgo.net/en/sorting?slide=1-2](https://visualgo.net/en/sorting?slide=1-2) Steven mentions a few array applications that become easier/more efficient if the underlying array is sorted. In this tutorial, we will discuss application 1-4 and if time permits, we will discuss application 5-6 (otherwise we will continue in the following week during Tut02).