CS2040C Semester 2 2021/2022
Data Structures and Algorithms

Tutorial+Lab 01
Basic C++, Basic OOP, Analysis, Hands-on 1
For Week 02 - early start due to no class in Week 04 (CNY Day 2)

Document is last modified on: January 12, 2022

1 Introduction and Objective

The purpose of this first tutorial+lab (e-)session (4 F2F and 4 E-Learn) is to recap the first four sessions of CS2040C: Introduction, basic C++ (esp those new C++17 features, std::vector, and basic OOP), basic analysis of algorithm, and to ensure that all students can code a simple C++17 program using their own computer at home (and/or laptop that he/she intends to use for Week 11 F2F Practical Exam). The first half of the session is generally the 'tutorial' part and the second half of the session is generally the 'hands-on/lab' part. The tutors will control the timings and they don't have to divide the sessions exactly by half. There will be a short break during the transition.

As this is the first session, we will do a quick ice breaking at the start of the session. Your TA is your main contact person of CS2040C related queries this semester. Only contact Dr Steven/Dr Alan for questions that you are sure that your TAs cannot (or has no privilege to) answer, e.g., questions about class policy, appeal for plagiarism verdict (if caught beyond reasonable doubt and you want to plead your case), etc.

To get the most out of the tutorial part of these sessions, please try out all the questions in the tutorial component 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.

Every week, you will try to solve one selected Kattis problem during the 'hands-on/lab' component. The tutors already know the selected Kattis problems for this semester. However, these selected problems will be revealed to you on the spot each week (if you happen to already solve it, then you are free to just leave the session or actually you can stay back to help your peers – you can learn more things by observing how others approached the problem that you have solved – possibly with just one way). Tutor will guide all students to get (near) Accepted solution for each problem. These problems are not graded but attempting them during the hands-on time (and possibly to fully complete them afterwards) is beneficial to better understand CS2040C material.

The tutorial/lab participation marks returns to encourage class participation. These marks will be given by the tutor **at the end of the semester** using the following guideline:

- 0% if you only attend ≤ 5 out of 11 tutorial/lab sessions (still 11 sessions as we replace Wed of Week 04 (CNY Day 2 PH) with Wed of Week 02),
- 1% for **at most the bottom three** most-passive students (assuming these students attend > 5 tutorial/lab sessions),
- 3% for **at least the top three** most-active students (answering questions when asked by TA – the correctness of your answers are secondary; or even just by asking your own questions to TA before/during/after class/during consultation); in each tutorial group, and
- 2% for the rest.

2 Tutorial 01 Questions

Q1). You are given a simple C++ program below:

```
#include <iostream> // question 1a
using namespace std; // question 1b
template<class T> // question 1c
class ListArray {
private:
  int N;
  T A[100]; // question 1d
public:
  ListArray() : N(0) {} // question 1e
  T get(int i) {
    return A[i]; // question 1f
  }
  int search(T v) {
    for (int i = 0; i < N; ++i)
      if (A[i] == v)
        return i;
    return -1;
  }
  void insert(int i, T v) {
    if ((N == 100) || (i < 0) || (i > N)) return; // question 1g
    for (int j = N-1; j >= i; j--)
    // for (int j = i; j <= N-1; ++j) // question 1h
      A[j+1] = A[j];
    A[i] = v;
    ++N;
  }
  void remove(int i) {
    for (int j = i; j < N-1; ++j) // question 1i
      A[j] = A[j+1];
    --N;
  }
  void printList() {
    for (int i = 0; i < N; ++i)
      cout << (i ? " " : "") << A[i]; // question 1j
    cout << '\n';
  }
  void sortList() { } // sort array A, question 1k
};
```

```
int main() {
  ListArray<int>* LA = new ListArray<int>(); // question 1l
  LA->insert(0, 5);
  LA->insert(0, 1);
  LA->insert(0, 4);
  LA->insert(0, 7);
  LA->insert(0, 2);
  LA->printList(); // we should have A = {2, 7, 4, 1, 5} by now
  cout << LA->get(3) << '\n'; // 1, A[3] = 1
  cout << LA->search(4) << '\n'; // 2, A[2] = 4
  cout << LA->search(6) << '\n'; // not found, -1
  LA->remove(1);
  cout << LA->search(4) << '\n'; // 1, A[1] = 4 now
  cout << LA->search(7) << '\n'; // not found, -1
  LA->printList(); // unsorted, A = {2, 4, 1, 5} by now
  LA->sortList();
  LA->printList(); // sorted, A = {1, 2, 4, 5} by now
  return 0;
} // please copy paste the code above, test compile, and run it yourself
```

This code will be revisited soon during discussion of List ADT (read `https://visualgo.net/en/list?slide=2-1` until 2-8). For now, please answer the following sub-questions (see the comments):

(a) Can we replace this line with `#include<bits/stdc++.h>`? What happened if we do so?

(b) Any potential issue with this line?

(c) What does this line means?

(d) Anything wrong with this line?

(e) What does this line means?

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

(g) What does this line means?

(h) What if we use this commented line instead of the line before it? Any potential issue?

(i) Any potential issue with this line?

(j) What does this line means?

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

(l) Can we just write `ListArray LA;` in this line?

Analysis/Order of Growth

Q2). What is the bound of the following function? $\mathbf{F}(n) = \log(\mathbf{2}^n) + \sqrt{n} + \mathbf{100\,000\,000}$

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

O(F(n)) = O(n log(2) + n^0.5 + 100 000 000)
= O(n + n^0.5) // remove constants
= O(n) // adding n with another n combines to n

In kattis, removing constants may not be a good idea cos it still adds to the time complexity

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

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

O(F(n)) = O(n + 1/2n + 1/3n .... + 1) = This is the harmonic series aka integration, area gets smaller and smaller towards infinity -> which gives a (log n) curve

O(G(n)) = In this series, the series is a(1-r^n)/(1-r) aka the Geometric series -> 2^n

Q3.b). What about $\mathbf{G}(n) = n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n + \ldots + \mathbf{1}$

Hands-on 1

TA will run the second half of this session with another chosen Kattis problem involving basic C++.

Problem Set 1

We will end the tutorial with discussion of PS1 A (/basicprogramming1) and PS1 B (/fluortanten) that will due this Saturday.

3 Note

Remember that outside the official tutorial+lab hours, all tutors will stand by at their designated consultation time slot + e-venue (Zoom session) each week (unless they mention some exceptions) to answer CS2040C related queries. This information can be found at `https://www.comp.nus.edu.sg/~stevenha/cs2040c.html`, scroll to 'registration' section. Remember that you do NOT have to be in that tutor's class to join a tutor's consultation slot. It is a free-and-easy unstructured session. Anyone who wants to study together with the tutor or have any CS2040C related question(s) can join.