NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

MIDTERM QUIZ FOR

Semester 1, AY2020/21

CS2040C – Data Structures and Algorithms

30 September 2020 Time allowed: 1.5 hours

STUDENT **NO. :**

INSTRUCTIONS TO CANDIDATES

1. Copy the `ans_blank.txt` text file and rename it to `<your NUSNET ID>.txt` e.g. `e0123456.txt`. Write your **student number** and **NUSNET ID** within the appropriate tags of the copied file. Ensure you **answer question 0** as instructed. Failure to do so will prevent your submission from being graded
2. This is an open-hardcopy-notes examination but **WITHOUT** electronic materials
3. It is your responsibility to ensure that you have submitted the correct file with the correct particulars and format. If you submit the wrong file, name the file incorrectly, fail to provide correct particulars or change the contents of the file such that it cannot be parsed, we will consider it as if you did not submit your answers. In the best case, marks will be deducted
4. No extra time will be given at the end of the test for you to write your particulars. You must do it **before** the end of the test. However, you may upload your file after the end of the test
5. This paper consists of **5** inline questions including Q0, and **2** multiline questions. It comprises **nine (9)** printed pages including this front page
6. Answer all questions within the text file. **Inline** answers should be appended to the end of each `#Qx:` part on the **same line**. Multiline answers should be **written between** the appropriate tags with **proper indentation** and adhering to the line limit. Avoid using the `#` character in both cases. Do NOT add, modify, remove any tag
7. Marks allocated to each question are indicated. Total marks for the paper is **100**
8. The use of electronic **calculator** is **NOT** allowed

Question	***Max***	***Marks***
Q0	Min -100	
Q1abcde	15	
Q2abcde	5	
Q3abcd	20	
Q4abcde	25	
Q5ab	10	
Q6	10	
Q7	15	
Total	**100**	

Question 0 **[0 for ENTIRE PAPER if not done satisfactorily!]**

Please read the following NUS Code of Student Conduct (in particular the part on Academic, Professional and Personal Integrity), as well as items B and C below.

(A) **I am aware of, and will abide by the NUS Code of Student Conduct (in particular the part on Academic, Professional and Personal Integrity as shown below) when attempting this assessment.**

- Academic, Professional and Personal Integrity
  1. The University is committed to nurturing an environment conducive for the exchange of ideas, advancement of knowledge and intellectual development. Academic honesty and integrity are essential conditions for the pursuit and acquisition of knowledge, and the University expects each student to maintain and uphold the highest standards of integrity and academic honesty at all times.
  2. The University takes a strict view of cheating in any form, deceptive fabrication, plagiarism and violation of intellectual property and copyright laws. Any student who is found to have engaged in such misconduct will be subject to disciplinary action by the University.
  3. It is important to note that all students share the responsibility of protecting the academic standards and reputation of the University. This responsibility can extend beyond each student's own conduct, and can include reporting incidents of suspected academic dishonesty through the appropriate channels. Students who have reasonable grounds to suspect academic dishonesty should raise their concerns directly to the relevant Head of Department, Dean of Faculty, Registrar, Vice Provost or Provost.

(B) **I have read and understood the rules of the assessments as stated below.**

1. Students should attempt the assessments on their own. There should be no discussions or communications, via face to face or communication devices, with any other person during the assessment.
2. Students should not reproduce any assessment materials, e.g. by photography, videography, screenshots, or copying down of questions, etc.

(C) **I understand that by breaching any of the rules above, I would have committed offences under clause 3(l) of the NUS Statute 6, Discipline with Respect to Students which is punishable with disciplinary action under clause 10 or clause 11 of the said statute.**

3) Any student who is alleged to have committed or attempted to commit, or caused or attempted to cause any other person to commit any of the following offences, may be subject to disciplinary proceedings:

l) plagiarism, giving or receiving unauthorised assistance in academic work, or other forms of academic dishonesty.

To answer Q0, type either your student number or NUSNET ID to declare that you have **read and will abide** by the NUS Code of Student Conduct (in particular, (a) Academic, Professional and Personal Integrity), (b) and (c).

Ans:

Question 1 **[15 marks, 5 x 3]**

<u>Without</u> running any C++ 17 compiler (you will be watched over Zoom), what are the outputs of the following C++ snippets (that share the same starting code template below). Variations of these C++ 17 code have been demonstrated throughout the first half of the course so far.

```
#include <bits/stdc++.h> // this is the standard template
using namespace std;
int main() {
  // code 1a, 1b, 1c, 1d, 1e are separately shown below, they are independent
  return 0;
}
```

a) Code 1a

```
  int steps = 0, n = 128;
  for (int i = 0; i < n; ++i)
    for (int j = n; j > 0; j /= 2)    8 times here
      ++steps;
  cout << steps << "\n";
```

Ans: 1024

b) Code 1b

```
  int x = 2;
  switch (x) {
    case 2:
    case 3:
      cout << "here";
    case 5:
      cout << "too";
      break;
    case 7:
      cout << "blank";
      break;
  }
  cout << "\n"; // the original paper accidentally missed this ';'
```

Ans: here too

c) Code 1c

```
  double ans = 0.0;
  vector<int> myList = {1, 3, 2, 4, 7};
  for (auto &v : myList)
    ans += v;
  cout << fixed << setprecision(7) << ans/(int)myList.size() << "\n";
```

Ans: 3.4000000

d) Code 1d bubble sort algo using ASCII code

```
string inputLine = "thisisCS2040C";
int m = (int)inputLine.size(); // m is the length of the string
for (int i = m-1; i > 0; --i)
  for (int j = 0; j < i; ++j)
    if (inputLine[j] > inputLine[j+1]) // hint: number < ALPHA < alpha
      swap(inputLine[j], inputLine[j+1]);
cout << inputLine << "\n";
```

Ans: 0024CCShiisst

e) Code 1e

```
string pangram = "thequickbrownfoxjumpsoverthelazydog";
// l = (int)pangram.size(); // l is the length of the string
sort(pangram.begin(), pangram.end());
cout << lower_bound(pangram.begin(), pangram.end(), 'f') -
        pangram.begin() << "\n";
```

Ans:

Hint: if you have not used lower_bound before (e.g., for PS1 A – basicprogramming2, task t = 1), here is the excerpt from https://en.cppreference.com/w/cpp/algorithm/lower_bound

```
template< class ForwardIt, class T >
ForwardIt lower_bound( ForwardIt first, ForwardIt last, const T& value );
```

Returns an iterator pointing to the first element in the range `[first, last)` that is *not less* than (i.e. greater or equal to) `value`, or `last` if no such element is found. *To be precise, the return value is the:* Iterator pointing to the first element that is *not less* than `value`, or `last` if no such element is found.

Question 2 **[5 marks, 5 x 1]**

a) What is the time complexity of the code shown in 1a? (Independent from the other parts) State the time complexity w.r.t. **n** as **n** is not necessarily 128 as in 1a.

O (O(nlogn))

b) What is the time complexity of the code shown in 1b? (Independent from the other parts)

O (O(1))

c) What is the time complexity of the code shown in 1c? (Independent from the other parts) But this time, let **n = (int)myList.size()**, state the time complexity w.r.t. **n** as **n** is not necessarily 5 as in 1c.

O (O(n))

d) What is the time complexity of the code shown in 1d? (Independent from the other parts) State the time complexity w.r.t. **m** as **m** is not necessarily 13 as in 1d.

O (O(m^2))

e) What is the time complexity of the code shown in 1e? (Independent from the other parts) State the time complexity w.r.t. **l** as **l** is not necessarily 35 as in 1e.

O (O(l*log(l))) sorting take l*log(l)

Question 3 **[20 marks, 3+7+5+5]**

a) You are given an array **A** of 7 integers: `{1, 2, 3, 4, 5, 6, 7}`.
There are 7! = 5040 possible permutations of **A**, e.g., `{1, 2, 3, 4, 5, 6, 7}`, `{1, 2, 3, 4, 5, 7, 6}`, …, `{7, 6, 5, 4, 3, 1, 2}`, `{7, 6, 5, 4, 3, 2, 1}`.
How many permutations of A causes this version of optimized bubble sort to do just 1 pass?
(Short explanation is optional but may be needed to give partial marks for wrong answer)

To avoid ambiguity, here is the version of "optimized Bubble Sort" as shown in VisuAlgo,

```
do
  swapped = false
  for i = 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap(leftElement, rightElement)
      swapped = true
while swapped
```

But written in C++

```
    int indexOfLastUnsortedElement = n;
    bool swapped;
    do {
      swapped = false;
      for (int i = 1; i <= indexOfLastUnsortedElement-1; ++i) // 1 loop=1 pass
        if (A[i-1] > A[i]) {
          swap(A[i-1], A[i]);
          swapped = true;
        }
      --indexOfLastUnsortedElement;
    }
    while (swapped);
```

Ans: 1

b) Same as above: You are given an array **A** of 7 integers: `{1, 2, 3, 4, 5, 6, 7}`.
How many permutations of A causes this version of optimized bubble sort to do at least 6 passes, and in the 6th pass, swaps are still being made?

To avoid doubt, permutation **A'** = {6, 2, 3, 4, 5, 1, 7} only need 5 (real) passes to get sorted with the 6th pass only to verify that there is no more swap, whereas permutation **A''** = {7, 6, 5, 4, 3, 2, 1} needs all 6 passes to get sorted.

(Short explanation is optional but may be needed to give partial marks for wrong answer)

Ans: 6!

c) RadixSort(A, d) has the following pseudo-code:

```
for i = 1 to d // digit 1/d is the least/most significant digit
  use a stable sort to sort array A on digit i // the inner loop
```

Someone (because he/she hasn't complete CS2040C) suggested that you should use merge sort in the inner loop of RadixSort pseudo-code above, as it is (usually) a stable sorting algorithm. There are two sub-questions:
Is the proposed suggestion algorithmically correct?
Will you use the proposed suggestion (why or why not)?

Ans:

d) I am a sorting algorithm.
I am good at sorting integers, big, (very) big (arbitrary precision), zeroes, small negative, very (small) negative etc.
Not just that, I can also sort *any* other data types as long as you tell me how to decide **a** less than **b** for any pair (**a**, **b**).
My worst-case processing speed is known to be "theoretically" optimal, sorting many integers is easy for me.
Additionally, I possess the stable sorting property.
Again, there are two sub-questions:
Who am I?
Btw, am I the most frequently used sorting algorithm among other competitors with similar speed (why or why not)?

Ans:
Mergesort. However, it is not the most frequently used as quicksort is O(nlogn) too but does it in place instead of requiring additional memory unlike mergesort

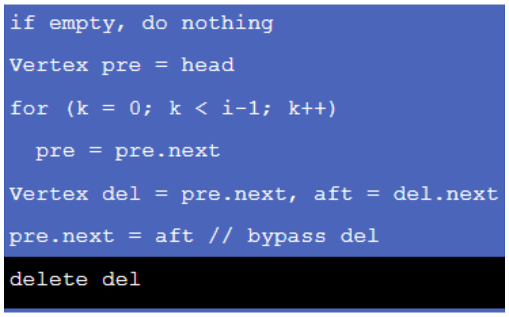
Question 4 **[25 marks, 5 x 5]**

a) In PS1 A – basicprogramming2, task t = 2, you were asked to detect whether the input array **A** "`Contains Duplicate`" or not ("`Unique`"). Obviously, it is unlikely that anyone uses a Linked List to store this input array **A**. For the purpose of this question, you will do exactly that. Given a **<u>(Singly) Linked List</u> L1** of **N** non-negative 32-bit signed integers (not necessarily sorted), determine if **L1** "`Contains Duplicate`" or not ("`Unique`"). Can you solve this task in O(**N**) or faster <u>without</u> using any additional data structure (i.e., process in place)?. If you can, briefly mention the idea. If you cannot, explain the reason succinctly.

Ans: No. You need to compare each element with all subsequent elements to check for duplicates. This requires nested loops, which result in quadratic time complexity.

b) You are given a (Singly) Linked List **L2** of N non-negative 32-bit signed integers (this time, it is guaranteed that **L2** is sorted in ascending order).

You are also given `Remove(i)` operation pseudo-code (in VisuAlgo) as follows that removes index i ∊ [1..**N**-2] and you call `Remove(i)` operation on **L2** that currently have **N** > 2 integers:



```
if empty, do nothing
Vertex pre = head
for (k = 0; k < i-1; k++)
  pre = pre.next
Vertex del = pre.next, aft = del.next
pre.next = aft // bypass del
delete del
```

Suppose the last line ("`delete del`") of that pseudo-code is commented out. What do you think will happen to SLL implementation that implements that pseudo-code (remember, **L2** is sorted)?

Ans: Nothing will happen. Subsequent operations on the SLL will work correctly.
However, del is not deallocated or freed -> memory leaks.

c) You want to implement a standard Stack ADT (push, peek, pop operations in O(1) as discussed in class). Which underlying data structure is the best (among the limited options below)? Why?
The limited options for this question are: [fixed-size array **A**, C++ std::deque **D**, or binary heap **H**]

Ans: Deque. Array requires size to be known in advance, binary heap does insertion and deletion on O(logN)

d) In this COVID-19 pandemic affected world, many businesses are implementing "social distancing queue" where customers/clients are (physically) queueing about 1 (or more) metre(s) apart and limiting total length of the queue to a maximum capacity **C**.

You want to implement this "social distancing queue" ADT as a computer program.
Which underlying data structure is the best (among the limited options below)? Why?
The limited options for this question are: [fixed-size array **A**, C++ std::deque **D**, or binary heap **H**]

Ans: Deque. Array not because removal of items from front position is slow due to the need to shift items. Binary heap insertion and deletion LogN

e) You have learned about standard queue ADT in class. The default version of a queue ADT is to support O(1) **enqueue** and O(1) **dequeue** operations for homogenous data type, e.g., queue of integers, queue of strings, etc.

Now, there are 2 types of data: strings and integers in your new "dual-type-queue". It supports 4 operations:

1. enqueue either a string OR an integer into your queue,
2. dequeueString(): return and dequeue the front-most string in your queue or report NULL otherwise,
3. dequeueInteger(): return and dequeue the front-most integer in your queue or report NULL otherwise, and
4. front() that will *print* the front-most *string or integer* (whichever comes first) currently in "dual-type-queue".

Can you implement these 4 "dual-type-queue" operations all in O(1)? If yes, show a simple idea on how to do that.

Ans: dual-type-queue consist of 2 separate queues and an index for every element.
Each queue will store the element, and the global index.
dequeue for both string and int dequeues from respective queues.
front() would peek at first element from both queues, and prints the one with the lower global in

Question 5 **[10 marks, 2 x 5]**

a) Show the compact array of size 6 (index 0 is empty, please put a -1 there) of a max Binary heap of 5 integers `{1, 2, 3, 4, 5}` so that when we do an `ExtractMax()` operation and the last existing leaf **X** is promoted to the root and we call `ShiftDown(1)`, this **X** **does NOT** trickle down to be another leaf again.

Ans: 5, 4, 1, 2, 3

b) Suppose that you have an ADT (Max) Priority Queue implementation that already supports O(log **n**) `Remove(v)` where `v` can be *any* key that exists in the Priority Queue (not necessarily the maximum element) on top of the standard O(log **n**) `Insert(v)` and O(log **n**) `ExtractMax()`. `Remove(v)` reports `NULL` if `v` does not exist in the Priority Queue. Show how to use just these subset of readily implemented ADT (Max) Priority Queue implementation to implement a "new" Priority Queue operation: `IncreaseKey(v, delta□v)` also in O(log **n**) where a key with value `v` (*nothing happens* if `v` does not exist in the Priority Queue) changes its value to `v+delta□v`. You can assume that the Priority Queue contains distinct keys.

Ans:

```
IncreaseKey(v, delta_v) {
Remove(v);
Insert(v+delta_v);
}
```

Question 6 **[10 marks]**

There is an array of **N** random floating-point numbers of unknown precision. However, it is also known that each floating-point number is at most **K** position(s) away from its position in the same array but with the numbers in sorted order (for this problem, $0 \le \mathbf{K} < \log \mathbf{N}$). **Propose the fastest (and correct) algorithm** (either in pseudocode/English or in full C++/no bonus marks for doing this other than to strongly signal your live coding skill to the lecturer) to sort the array and **analyze its time complexity**.

You will get 0, 2, 3, 5, or 10 marks if your answer is wrong (max 1 partial mark for effort), "correct but O($\mathbf{N}^2$)", "correct but O(**N** log **N**)", "correct but O(**NK**)", or "correct and better time complexity than the other options", respectively. nlogk

Example: **N** = 5, **K** = 3, {1.618…, 0.33…., 3.1415…, 0.66…, 2.718…} → {0.33…., 0.66…, 1.618…, 2.718…, 3.1415…}

[Line limit **15**] Ans:

MULTILINE
ANSWER

Question 7 **[15 marks]**

A group of **N** IOI students are sitting in a circle. Each IOI student has a skill level given as an integer [0..10^9] with the top student has the highest skill (yes, a very fine grained differences matter at this level). Steven, who is both the IOI leader of team SGP (on the years where SGP is not hosting the IOI) and coordinator of the NUS ICPC team wants to pick **K** *consecutive* students to form an ICPC team (usually **K** = 3, but for the purpose of this question, 1 ≤ **K** ≤ **N**).

From his vast experience, Steven knows that the skill level of the team can be approximated by the skill of the student with the weakest skill level (i.e., "a chain is no stronger than its weakest link"). So the question, what is the highest possible skill level of the team that can be formed?

Propose a correct algorithm (either in pseudocode/English or in full C++/no bonus marks for doing this other than to strongly signal your live coding skill to the lecturer) to do this and **analyze its time complexity**.

You will get 0, 5, 10, or 15 marks if your answer is wrong (max 2 partial marks for effort), "correct but O(**NK**)", "correct but O(**N** log **K**)", or "correct and better time complexity than the other options", respectively.

Example, **N** = 5, **A** = {5, 7, 3, 10, 4}, **K** = 3, then among the 5 possible teams of 3 consecutive IOI students (with the weakest skill level per team underlined):

A. {5, 7, 3},
B. {7, 3, 10},
C. {3, 10, 4},
D. {10, 4, 5} (loop around),
E. {4, 5, 7} (loop around).

The highest possible skill level of the team that can be formed is 4 (either team D or team E).

[Line limit **25**] Ans:

MULTILINE
ANSWER

- End of Paper -