National University of Singapore

School of Computing

CS2040C - Data Structures and Algorithms
Midterm Test

(Wed, 3 Oct 2018, S1 AY2018/19, 80m)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains FIVE (5) sections.
   It comprises TEN (10) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed spaces** in this booklet.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   For full marks, you must use **C++ code** but beware of penalty marks for **compilation error**.
   You can use **standard, non-modified** algorithm discussed in class by just mentioning its name.
7. Write your Student Number in the box below:

A	0	1						

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks
A	10		
B	15		
C	15		
D	40		
E	5 (Bonus)		
Total	Max 80		

A Worst Case Time Complexity Analysis (10 marks)

Write down the *tightest*[1] *worst case* time complexity of the various data structure operations or algorithms below.

The operations (algorithms) referred to below are the **unmodified version**, as per discussion in class or as currently implemented in C++ STL. **Unless otherwise mentioned, there are currently n elements in the data structure**, which can be in any arbitrary order.

No	Operations		Time Complexities
1	Obtain the size of a `std::vector` using `.size()`.		$O(\_\_\_\_\_\_\_)$
2	Check if two `std::string` are equal.		$O(\_\_\_\_\_\_\_)$
3	Insert a new element into a **sorted** `std::vector`.		$O(\_\_\_\_\_\_\_)$
4	Run *non-randomized* **Quicksort** on a *sorted* array.		$O(\_\_\_\_\_\_\_)$
5	Run **Mergesort** on the *first half* of an array (first $\frac{n}{2}$ elements).		$O(\_\_\_\_\_\_\_)$
6	Check if a **Singly Linked List** is in *sorted order.*		$O(\_\_\_\_\_\_\_)$
7	Swap the *first* and *last* elements of a **Doubly Linked List**.		$O(\_\_\_\_\_\_\_)$
8	Get the *bottommost* element of `std::stack`.		$O(\_\_\_\_\_\_\_)$
9	Reverse the current content of `std::queue`.		$O(\_\_\_\_\_\_\_)$
10	Remove the *first* and *last* elements of a `std::deque`.		$O(\_\_\_\_\_\_\_)$

[1] What we mean by tightest worst case time complexity is as follows: If an operation of the stipulated data structure/an algorithm needs at best $O(n^3)$ if given the worst possible input but you answer higher time complexities than that, e.g. $O(n^4)$ – which technically also upperbounds $O(n^3)$, you will get wrong answer for this question.

Section A Marks = _____

B Analysis (15 marks)

Prove (the statement is correct) or disprove (the statement is wrong) the statements below.

1. When writing a C++ `class`, it is possible to not implement/define any constructors and still be able to compile the program.

2. There is **no** sorting algorithm that is **both** *in-place* and *stable*. (You do not have to define in-place or stable in your responses).

3. In Mergesort, the time complexity is $O(n \log_2 n)$ as there is a total of $\log_2 n$ layers when we split the array into 2 at each *divide step*. If we instead divide the array into n parts of 1 element each, we can improve Mergesort to run in $O(n \log_n n)$ which simplifies to $O(n)$.

4. Insertion Sort will *never* run faster than Mergesort when they are used to sort the same array.

5. It is possible to print/output a Singly Linked List of n integers in reversed order with $O(n)$ time complexity.

Section B Marks = _____

C Create Test Cases (15 marks)

Create Test Cases for each scenario below. Each valid test case is worth 5 marks.

1. Supply an input array `X` of `N = 7` **distinct** integers ∈ [1..7] that is the *best* input case for *(non-randomized)* Quicksort. Assume that Quicksort always sets the pivot to the element with the lowest index of the partition. It should take **exactly 10 (pairwise) comparisons** for Quicksort to make `X` sorted in increasing order.

2. Supply an input array `X` containing *any number* of **distinct** integers ∈ [100..999] such that **Selection Sort** will take more **loop iterations** than **Radix Sort** to make `X` sorted in increasing order. Recall that/or assume Radix Sort looks at integers digit by digit and splits them into 10 groups at each phase. (Hint: What are the time complexities of Radix Sort and Selection Sort?)

3. Supply an input array `X` of `N = 9` integers such that the output is the same for both of the code snippets below. Your score for this scenario will be the number of **distinct** integers in `X`, capped at 5.

Snippet A	**Snippet B**
`stack<int> s;`	`queue<int> q;`
`for (int i = 0; i < 9; i++) {`	`for (int i = 0; i < 9; i++) {`
`  int x;`	`  int x;`
`  cin >> x;`	`  cin >> x;`
`  s.push(x);`	`  q.push(x);`
`}`	`}`
`for (int i = 0; i < 9; i++) {`	`for (int i = 0; i < 9; i++) {`
`  cout << s.top() << endl;`	`  cout << q.front() << endl;`
`  s.pop();`	`  q.pop();`
`}`	`}`

Section C Marks = _____

D Applications (40 marks)

D.1 Groupings (25 marks)

Prof. Gary has a list of N students and their respective scores in a particular exam. The scores range from 0 to 10^9 inclusively. In addition, there can be up to 100 000 students but no less than 2 students. Out of these N students, Prof. Gary wants to select exactly K students ($2 \leq K \leq N$) to form a group for a special project. However, he wants the *difference* between the highest scoring student and the lowest scoring student in the group to be **minimized**. Your job is to help Prof. Gary compute the **minimum** difference in highest and lowest scores among all the possible groups of K students he can form. You **do not** need to output the set of K students that should be selected.
The input has 2 integers in the first line: N followed by K. The second line will contain N integers, representing the scores of the students.

Sample Input	$\rightarrow$	Sample Output
10 4 0 726 938 572 746 256 512 1020 726 2040		174 (Selected students: 726, 572, 746, 726)

Special Case 1 (5 marks): What is the answer when $K = N$? Using **words**, describe the most efficient algorithm that will solve this special case. State the time complexity of your algorithm.

The time complexity of my algorithm is O(________).

Special Case 2 (5 marks): What is the answer when $K = 2$? Using **words**, describe the most efficient algorithm that will solve this special case. State the time complexity of your algorithm.

The time complexity of my algorithm is O(________).

General Case (15 marks): Design an efficient algorithm to solve the general case of this problem for all values of K between 2 and N. Do include the time complexity analysis for your code.
You have to complete this question using C/C++ code. If you are not sure of the required C++ syntax, you can write your answer in pseudo-code for slightly lesser marks.

```
#include <bits/stdc++.h>
using namespace std;







int main() {
  int N, K;
  cin >> N;   //N is the number of students
  cin >> K;   //K is the number of students to be selected to form a group
  int scores[N];   //scores is the array of student scores
  for (int i = 0; i < N; i++) {
    cin >> scores[i];    //scores[i] ranges from 0 to 1 billion inclusive
  }














  return 0;
} // the overall time complexity of my C++ code above is O(_____________)
```

D.2 Bus Rides (15 marks)

Rar the Cat is driving a bus with only one door. However, the bus is too narrow for people to rearrange themselves in the bus. Hence, it operates in a **First-In-Last-Out** manner where the first person to enter the bus needs to wait for everyone else that entered after him/her to exit before he/she can do so. For instance, if person A enters the bus then followed by persons B and C, then person C must be the first one to exit, followed by B and then A.

The bus is driven along a long road with N bus stops, labelled from 1 to N. The bus goes to the bus stops in **increasing order**. (i.e. it will go to bus stop 1, then 2, 3, ..., N). Along the road, there are P people waiting for a bus ride, labelled from 1 to P. Person i is currently at bus stop S_i and wants to get to bus stop D_i. It is guaranteed that there will not be more than one person waiting for a bus ride at any bus stop. In addition, there will not be more than one person that wants to get off at the same bus stop. In other words, S_i is distinct for all P people and D_i is distinct for all P people as well. If a person wants to alight at a bus stop, and another person wants to board from the same bus stop, it is assumed the person on the bus will alight first.

Rar the Cat wants to know if the bus can be used to transport all the people from their starting locations to their destinations. If it is possible, output "Yes", otherwise output "No". Recall that the bus only moves from bus stops 1 to N **once** in increasing order.

The input has 2 integers in the first line: N followed by P. N ranges from 1 to 1 000 000 inclusive and P ranges from 0 to N. P lines of input will follow, the i^{th} line will contain integers S_i then D_i, representing that person i wants to get from bus stop S_i to D_i. S_i and D_i will range from 1 to N.

Input	→	Out	Explanation
12 4	→	Yes	Pick up person 1 from stop 1, person 4 from stop 2.
1 12			Drop off person 4 at stop 5, pick up person 3 at stop 5.
6 9			Pick up person 2 at stop 6 and drop him off at stop 9.
5 10			Drop off person 3 at stop 10.
2 5			Drop off person 1 at stop 12.
11 4	→	No	The bus cannot go from stop 5 to stop 3.
1 9			Hence, person 3 cannot be transported.
2 10			Person 1 and 2 cannot be transported together.
5 3			Person 1 needs to board before person 2, but alight before person 2.
3 7			Bus is First In Last Out, this is not possible.

Quick Question (1 mark): Which **Linear Data Structure** does the bus most resemble?

Actual Question (14 mark): Design an efficient algorithm to solve this problem on the next page. Do include the time complexity analysis for your code.

You have to solve this question using C/C++ code, including input and output. If you are not sure of the required C++ syntax, you can write your answer in pseudo-code for slightly lesser marks.

```
#include <bits/stdc++.h>
using namespace std;







int main() {

























  return 0;
} // the overall time complexity of my C++ code above is O(______________)
```

Section D Marks = _____ + _____ = _____

E Easy Marks (5 bonus marks)

Write a **short** (limit yourself to just 2-3 minutes to do this) **but honest** feedback on what you have experienced in the first 6 weeks of CS2040C. Suggestions that are shared by majority (not a one-off feedback) and can be easily incorporated to make the next 6 weeks of CS2040C better (Week 07-12) will be considered. Grading scheme: 0-blank, 1/2-considered trivial feedback but not blank, 4/5-good and constructive feedback, thanks. These bonus marks will be added to your score but the maximum score for this paper remains at 80 marks. (In other words, you cannot get higher than 80/80.)

– End of this Paper, All the Best –

Blank paper: Can serve as rough paper or extra response space (please ***clearly state which question****)*