National University of Singapore

School of Computing

CS2040C - Data Structures and Algorithms Midterm Test

(S2 AY2025/26; Tue, 03 March 2026)

Time Allowed: 80 minutes

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections.
   It comprises EIGHT (8) printed pages, including this page.
3. This is an **Open Book Assessment**.
   You cannot use any electronic device except one calculator.
4. 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 (sub-)questions might be easier than they appear.
6. Section B of this paper has special format.
   Read the instructions carefully.
7. The total marks is 50. All the best :)

A Midterm Feedback ($2 \times 1 = 2$ marks)

Prof Halim returns to teach CS2040C in S2 AY2025/26 after focusing on CS2040S for the last 3 AYs. There may be differences of style/expectation between CS students versus CEG/InfoSec/Exchange students. There is also that non-ideal situation involving Prof Halim's personal circumstances.

Write two sentences about the first half of CS2040C of S2 AY2025/26.
The first sentence is about what you like the most about the course so far.
The second sentence is about what you do not like the most and hope for change(s) in the second half.

B Application Questions ($3 \times 16 = 48$ marks)

Suppose for all three application questions below, you copy-pasted the entire problem statement and asked your Generative AI friend 'talkTwinSeek' to generate C++ solutions for you. This GenAI generates three C++ code that are shown in the answer sheet (assume that all necessary includes, namespace std, and the main function are added automatically).

For all three application problems of this special midterm paper, your task is to check 'talkTwinSeek' outputs. There are a few possibilities and there is a custom marking scheme depending on how accurate is your analysis and/or (updated) solutions:

AC). If you think the generated C++ code is 100% correct (AC), just declare that it is correct.

WA). If you think the generated C++ code is wrong (WA), identify the bug(s), explain why it is/they are buggy, and suggest as minimal fix as possible to the buggy part(s).

TLE). If you think the generated C++ code is correct but is (too) slow (TLE), identify which part(s) is/are slow - analyze this slow time complexity, rewrite those part(s) with the faster solution(s), and re-analyze the faster time complexity. Note that in class/programming assignments, we generally say $\Omega(n^2)$ — i.e., n^2 or worse – is too slow and $O(n \log n)$ — i.e., $n \log n$ or faster – is needed, but if the generated code is $O(\log n)$ and you know the $O(1)$ solution, you should still suggest this improvement.

RTE). If you think the generated C++ code is generally fast enough and is the correct algorithm, but it has not handled specific corner case(s) that can cause the code to crash (RTE), identify such corner case(s), and suggest as minimal fix as possible to the buggy part(s).

MLE). If you think the generated C++ code uses too much memory. Analyze this big space complexity, rewrite the required part(s), then re-analyze the smaller space complexity.

Note that the generated code can be WA, TLE, RTE, and/or MLE at the same time. In this case, you may actually want to rewrite the whole thing. You are allowed to use **pseudo-code** instead of C++. But beware of penalty marks for **ambiguous answer**. As usual, you can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g. run Merge Sort on list L, use Stack S, etc. However, if your minimal fix is minor, you are allowed to just annotate directly in the generated C++ code.

Remember to (re)-analyze the tightest worst-case time complexity of the fixed solution, if any.

B.1 Hail Mary's Problem (16 marks)

The Eridian language that Rocky uses - after Dr Ryland Grace translate it with Hail Mary's computer - happens to also use English lowercase letters ['a'..'z'], but in a different order, i.e., the order of the alphabet of Eridian language is some permutation of lowercase letters.

Given a list of n ($n \geq 1$) `words` spoken by Rocky the Eridian or Grace the Human, and the `order` of the alphabet (exactly of length 26), return `True` if and only if the given words are sorted *lexicographically* in the specified language (otherwise, return `False`). The length of each word is between 1 to 21 characters. All characters in each word in `words` and in `order` are lowercase ['a'..'z'].

To help you solve this problem, consider the following exchange between Rocky and Grace:

- Input: `words = ["you", "where", "question"]`, `order = "ybcfwdqazxletrpknmhgijousv"`
  Output: `True`
  Explanation: `'y'` < `'w'` < `'q'` in Eridian language, so the list is sorted.

- Input: `words = ["i", "dont", "understand"]`, `order = "abcdefghijklmnopqrstuvwxyz"`
  Output: `False`
  Explanation: In normal English, the sorted order should be `["dont", "i", "understand"]`.

- Input: `words = ["you", "come", "from", "where", "question"]`,
  `order = ybcfwdqazxletrpknmhgijousv"`
  Output: `True`

- Input: `words = ["oh", "sol", "star"]`, `order = "abcdefghijklmnopqrstuvwxyz"`
  Output: `True`

- Input: `words = ["solaris", "sol"]`, `order = "abcdefghijklmnopqrstuvwxyz"`
  Output: `False`
  Explanation: `"solaris"` is longer than `"sol"`, the sorted order should be the other way around.

Solve this problem in C++ and analyze its time complexity.

B.2 It Takes Two (16 marks)

A certain big company has two equal-power bosses. To execute a new company policy, both bosses have to meet in person (secret meeting) and discuss the delicate matter between the two of them.

You are given two integer arrays `free1` (of $n > 1$ slots) and `free2` (of $m > 1$ slots) that represent the available free slots for both bosses. Each time slot is represented as a pair of integers {`start, end`}, where `start` and `end` are inclusive boundaries (meaning this boss is available from time `start` to time `end`, inclusive). The unit is in minutes and you can assume that $0 \leq start < end < 24 \cdot 60$. Note that within each boss's schedule, time slots don't overlap, i.e., for any two free time slots {`start1, end1`} and {`start2, end2`} of the same boss, either `start1 > end2` or `start2 > end1`.

You are also given an integer `duration`, representing the (estimated) length (in minutes) of a meeting that the two bosses need to discuss the new company policy.

You are the secretary of both bosses. Your task is to find the *earliest* time slot where both bosses are available for at least `duration` minutes. If such a common time slot exists, return an integer pair {`meetingStart, meetingEnd`} where the meeting would start at `meetingStart` and end at `meetingEnd = meetingStart+duration`. If there is no solution, return a special pair {`-1, -1`}.

Example: If
boss 1 has `free1 = [{610, 650}, {740, 810}, {660, 720}]`,
boss 2 has `free2 = [{660, 670}, {600, 615}]`, and
`duration = 8`, then the earliest common available time is {`660, 668`}.
However, if `duration = 11` (for the same `free1` and `free2`), then the answer is {`-1, -1`}.

Solve this problem in C++ and analyze its time complexity.

B.3 Split the SLL (16 marks)

Given the `head` of a Singly Linked List (SLL) of integers (between -99 to 99) and an integer k, split the SLL into k consecutive SLL part(s). Assume that there are n ($n \geq 0$) integer(s) in the SLL and $1 \leq k \leq n$, the length of each SLL part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being `NULL`. The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later. Return an array of the k parts. See an example in Figure 1.

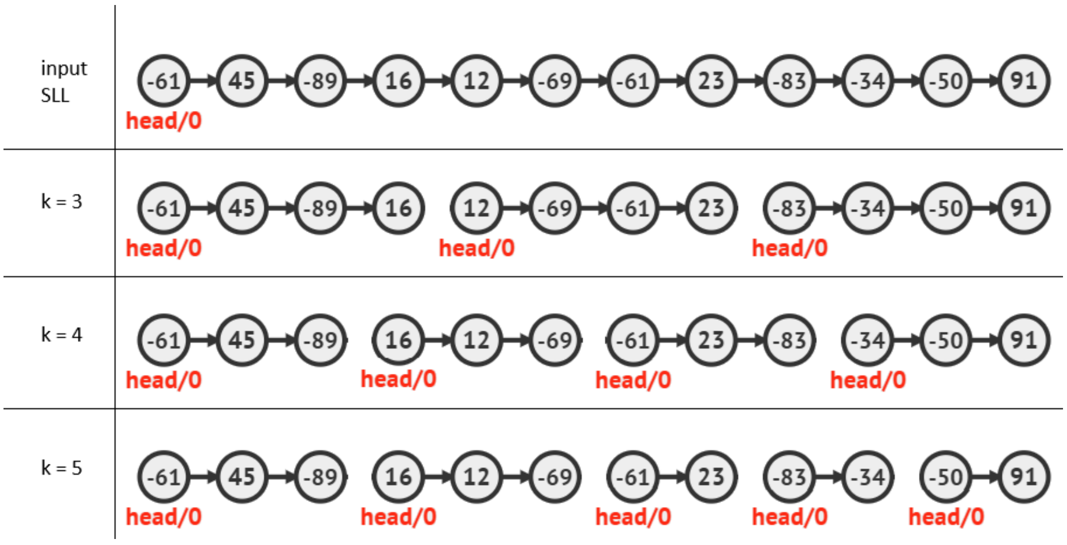
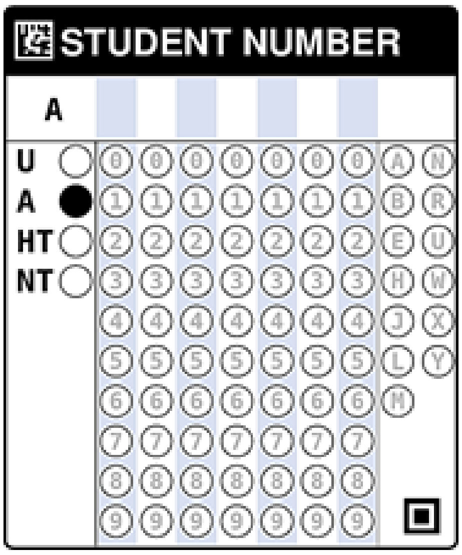


Figure 1: An example SLL (top) with $n = 12$ integers and a few splitting scenarios with different k

Solve this problem in C++ and analyze its time complexity.

The Answer Sheet for Semester 2 AY2025/26

Write your Student Number in the box below using **(2B) pencil**. **Do NOT write your name**.



Section	Maximum Marks	Your Marks	Grading Remarks
Total	50		

Box A.1. Feedback about CS2040C S2 AY2025/26:

What do you like the most?

What you do not like the most?

Box B.1. Hail Mary's Problem

```
class Solution {
public:
    bool is_sorted(vector<string>& words, string order) {
        vector<string> sorted_words = words;
        sort(sorted_words.begin(), sorted_words.end(), [&order](string a, string b) {
            // return a bool value indicating whether the first argument (string a)
            // should be placed before the second argument (string b)
            // in the sorted order (given in parameter 'order')
            int i = 0, x = (int)a.size(), j = 0, y = (int)b.size();
            while (i < x and j < y) {
                int o1 = order.find(a[i++]), o2 = order.find(b[j++]);
                if (o1 < o2)
                    return true;
                else if (o1 > o2)
                    return false;
            } // ps: variable names changed to x/y (size of string a/b, respectively)
            return (i < x) ? true : false;
        });
        return sorted_words == words;
    }
};
```

My analysis of the generated code above is:

That solution runs in $O($____________________$)$.

Box B.2. It Takes Two

```
// for pair<int, int>, .first and .second refers to
class Solution { // the first and second element of the pair, respectively
public:
    pair<int, int> earliestMeeting(vector<pair<int, int>>& free1,
        vector<pair<int, int>>& free2, int duration) {
        int bestStart = 1001;
        for (int i = 0; i < (int)free1.size(); ++i)
            for (int j = i; j < (int)free2.size(); ++j) {
                int start = max(free1[i].first, free2[j].first);
                int end = min(free1[i].second, free2[j].second);
                if (end-start >= duration)
                    bestStart = min(bestStart, start);
            }
        if (bestStart == 1001)
            return {-1, -1};
        return {bestStart, bestStart+duration};
    }
};
```

My analysis of the generated code above is:

That solution runs in $O($____________________$)$.

Box B.3. Split the SLL

```
class Solution {
public:
    vector<ListNode*> split_SLL(ListNode* head, int k) {
        int n = 0;
        ListNode* cur = head;
        while (cur != NULL) {
            ++n;
            cur = cur->next;
        }
        int size_per_part = n/k;
        vector<ListNode*> ans(k, NULL);
        cur = head;
        for (int i = 0; i < k; ++i) {
            ans[i] = cur;
            int cur_part_sz = size_per_part;
            while (cur_part_sz--)
                cur = cur->next;
            if (cur != NULL) {
                ListNode* temp = cur->next;
                cur->next = NULL;
                cur = temp;
            }
        }
        return ans;
    }
};
```

My analysis of the generated code above is:

That solution runs in $O(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)$.

– END OF PAPER; All the Best –