CS3219 AY2324 S1 Assignments

Maximum marks – 33
Submit by - 15 Nov Wednesday, 5pm
You are encouraged to complete these assignments earlier as they form the base for the course project.

There are six assignments listed below totaling 35 points.
The max score is capped at 33 points.
If you have scored full marks on Assignments 1-5, you can opt not to do Assignment 6.
Alternatively, you could submit all assignments and use Assignment-6 marks to compensate for lost marks on any other assignment.

These assignments have two-fold purposes. First, these scaffolded assignments allow you to learn and use some of the industry-relevant tools and technologies. Second, the outcome of these tasks provides you with a base for the course project. You complete these assignments in your respective project groups. Only one submission per group is expected for each assignment. Project group members can decide to distribute workload based on their skill set and interests. It is expected that each member will equally contribute. Project groups are expected to keep records of work allocations and actual contributions. Above record, Mentor's feedback and a peer review shall be considered in case there is a conflict.

We expect you to test the robustness of your assignment submissions – ensure you fortify them through adept error handling and meticulous exploration of diverse edge cases.

Important Note: *Please be aware that not adhering to the specified submission instructions may result in penalties of up to 20% of the maximum marks for these assignments. This includes instances such as submitting a video file that does not conform to the prescribed format, placing files in incorrect folders, using incorrect repository tags, or encountering repository accessibility issues due to specific settings.*

For any queries post to MS-Teams channel Clarifications [Assignments] or reach out to your project mentor.

Table of Contents

Assignment 1 (5 points)

Skills Developed

- Developing Single Page Application
- Client-side storage and local data persistence
- CRUD in JavaScript
- HTML + CSS

In this assignment, your objective is to build a web application tailored to manage a repository of questions within a question bank. Your task involves developing a Single Page Application (SPA) that can perform CRUD operations. That is, your application should support the addition of new questions, viewing individual questions, listing all the questions stored, and deleting questions. Each question must have the following details/format:

Question Id	Question Title	Question Description	Question Category	Question Complexity

Guidelines and Assumptions

- The web application will be hosted on your laptop; it does not need to be accessed from the internet.
- You will design a SPA (Single Page Application), where all the functionality is written within a single web page instead of having cross-references across many pages.
- The input questions data will be maintained within the browser using HTML, JS constructs. Use of database is not required in this task. (Note: you will implement the database for this application in Assignment 2)
    - One way to store data is using JavaScript cookies.
    - Another way is to use the local storage provided by browsers.
- For this assignment, you can enter the questions manually. You are allowed to have alternative methods of entering questions.
- You are given a sample of 20 questions as your input data that you can use to develop, test, and demonstrate your assignment. (See Appendix 3)
- You are required to show a list of questions (mandatorily show the title, optionally show the complexity/category or both) in the current state of the application on accessing the application (landing page). The question description should be shown on clicking the question title.
    - E.g., You can organize the questions in the form of a table on the SPA.
- Use CSS to make the page usable in terms of displaying the information in an organized manner.

Required Elements in Assignment 1 and Grading Rubrics

Requirement	**Points**
A landing page which can display basic information about all the questions in the current state of the application (e.g., title, complexity).	0.5
Functionality to add question with the aforementioned attributes.	0.5
Functionality to display details of each question on selecting (e.g., by clicking) the question.	1
Functionality to delete a question.	0.5
Use style sheets to make the page usable.	1
Persist data using JS cookies, localStorage or any other mechanism that you deem appropriate.	1
Error handling (e.g., basic error handling like checking for duplicate questions).	0.5

Submission Instructions

- Refer to Appendix 1 to create your team in GitHub Classroom.
- Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up.
- Tag the commit on the master branch that marks the end of Assignment 1 as "Assignment 1" and push the tags.
- Record a video demo showing the required elements mentioned above (time limit 3 minutes). Label the .mp4 file <your project group number>_Assignment-1 (e.g., G04_Assignment-1.mp4).
- Submit the video on Canvas-> Assignments -> Assignment-1.
- Note: The assignment marker (from teaching team) would call your group in case there is a need for clarification.

FAQ

Can we implement a database in this assignment and count it towards Assignment 2?

- ➢ Yes. Note: Assignment 2 requires you to develop a proper backend.

Assignment 2 (10 points)

Skills Developed

- REST API development
- Front-end & Back-end development
- SQL and NoSQL Databases, (Technologies: e.g., MongoDB/MySQL/Postgres)
- (Potential technologies/frameworks: React, NodeJS, Python)

In this assignment, you will learn how to create a backend for web applications using two different kinds of databases (SQL and NoSQL) and establish communication between the front-end and backend using REST.

There are two parts to this assignment:
(a) In this part, your objective involves creating and managing user profiles for any web application. You are required to implement core user actions like registering (C), viewing profiles (R), updating profiles (U), and deregistering from the platform (D). This will necessitate the establishment of a REST API for communication between the front-end and back-end, and data storage within an SQL database e.g., MySQL/Postgres.

(b) In this part, you will enhance Question Repository created in Assignment 1 by designing APIs for the CRUD operations and maintaining question records in a NoSQL database such as MongoDB.

Guidelines and Assumptions

You can use any programming language/platform for developing the REST API

Required Elements in Assignment 2 and Grading Rubrics

Requirement	**Points**
One point for demonstrating each of the CRUD operations, including robustness, for user profile management.	4
FE-BE integration for user profile management	3
Backend + integration with front end of the Question repository from Assignment 1	3

Submission instructions

- Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up.
- Tag the commit on the master branch that marks the end of Assignment 2 as “Assignment 2” and push the tags.
- Record a video demo:
    - Showing the CRUD operations for user profile from the front-end. Correspondingly show the updates made to the backend database.
    - Showing the integration of the backend with the Question repository frontend.
    - Duration of video – less than or about 3 minutes.
    - Label the .mp4 file <your project group number>_Assignment-2 (e.g., G04_Assignment-2.mp4)
    - Submit the video on Canvas-> Assignments -> Assignment-2

Note: The assignment marker (from teaching team) would call your group in case there is a need for a clarification.

Assignment 3 (5 points)

Skills Developed

- Authentication state management with JWT/session tokens
- Using OAuth or such 3rd party authorization protocol/services.

In this task, you will implement authentication and authorization functionality and integrating the question repository from Assignment 1 & 2, and user profile management from Assignment 2 using appropriate API.
Note: In Assignment 4 you will be containerizing each of the applications independently. Keep this in view while working on this assignment.

For auth functionality: Define roles among the users who can access the questions.

a. A designated maintainer role whose privileges allow users to update and delete the questions in the question repository.
b. Registered users should be allowed to read the questions from the question repository.
c. Unauthenticated and unauthorized users should not be allowed to access questions (either for reading or updating).

Guidelines and Assumptions

- This Assignment assumes you have completed Assignment 1 and Assignment 2.
- You can choose to develop your own authentication mechanism or use any third-party OAuth provider.

Required Elements in Assignment 3 and Grading Rubrics

Requirement	**Points**
Demonstrating user authentication for accessing questions	1
Demonstrating session management (e.g., on page refresh)	1
Demonstrating authorization management using roles using the above-mentioned roles	3

Submission instructions

- Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up.
- Tag the commit on the master branch that marks the end of Assignment 3 as “Assignment 3” and push the tags.
- Record a video demo:
    - Demonstrating that only authenticated users are able to access the questions.
    - Demonstrating session management in a few scenarios.
    - Demonstrating the authorization of users using roles to add and update the questions.
    - Demonstrating the HTTP failure messages in each of the above cases.
    - Duration of video – less than or about 3 minutes.
    - Label the .mp4 file <your project group number>_Assignment-3 (e.g., G04_Assignment-3.mp4).
    - Submit the video on Canvas-> Assignments -> Assignment-3.

Note: The assignment marker (from the teaching team) would call your group in case there is a need for a clarification.

Assignment 4 (3 points)

Skills Developed

- Containerization (Technology: Docker/Podman, Docker engine)
- Code integration

In this assignment, you will containerize the applications and functionalities developed from Assignment 1 through Assignment 3. You will write the Docker files necessary to containerize the applications.

Guidelines and Assumptions

- You can package the database in the same container as the respective application or choose to containerize the applications and databases separately.

Required Elements in Assignment 4 and Grading Rubrics

Requirement	**Points**
Containerize question repository application	1.5
Containerize the user profile management application	1.5

Submission instructions

- Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up.
- Tag the commit on the master branch that marks the end of Assignment 4 as "Assignment 4" and push the tags.
  - Ensure to include the relevant docker/docker compose files in your repository and tag the relevant commits.
- Record the demonstration of running the two applications out of the containers.
  - Label the .mp4 file <your project group number>_Assignment-4 (e.g., G04_Assignment-4.mp4).
  - Time limit of the video: 2-3 minutes.
  - Submit it on Canvas-> Assignments -> Assignment-4.

Note: The assignment marker(from teaching team) would call your group in case there is a need for a clarification.

Assignment 5 (10 points)

Skills Developed

- Developing Front-end/view layer
- Using queuing technologies (e.g., one of JMQ/MSMQ/AMQP)

In this assignment, you will develop an application to provide functionality for initiating a collaboration between two users based on some criteria (e.g., working together on a question of a particular difficulty level).

Guidelines and Assumptions

- Two users should be able to specify the difficulty level of the question they want to work on.
  - You can also consider including the question category as a criterion for matching in addition to the difficulty.
  - Note: You can consider integrating with the user profile management application developed in Assignment 2 OR create a separate set of users to demonstrate this application.
- Each user can wait a fixed amount of time (say 30 seconds) to find a match.
- Provide some form of feedback to the user using the fronted code (e.g., animation to indicate the user to wait until match is obtained)
- You can handle the scenario where no match is found in various ways (e.g., retry matching, cancel the matching attempt with an error, etc.,)
- You will typically use some form of queuing mechanism to achieve the matching.
- You are required to containerize this application like the previous ones.

Required Elements in Assignment 5 and Grading Rubrics

Requirement	**Points**
Demonstrating the two users' inputs to specify the criteria for matching	2
Front-end timer + feedback + handling no match	4
Demonstrating the valid matching	2
Containerizing the application	2

FAQ

How do we demonstrate the valid matching?

➢ Show the state of the application/state of the queue to indicate a match was found.

Submission instructions

- Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up.
- Tag the commit on the master branch that marks the end of Assignment 5 as "Assignment 5" and push the tags.
  - Ensure to include the relevant docker/docker compose files in your repository and tag the relevant commits.
- Create a PDF file with the following to submit on Canvas.
  - Screenshot of the view layer/front end showing
    - The timer
    - Feedback message/updated UI on successful match
    - Feedback message on unsuccessful match

    - Debug/log message showing indicating the status of the queue before and after match.
    - Name the file <your project group number>_Assignment-5.pdf (e.g., G04_Assignment-5.pdf)
    - Submit the PDF to Canvas -> Assignments -> Assignment-5
- **Note: this will be an in-person demonstration. Arrange a slot with your project mentor once the assignment is complete.**

Assignment 6 (2 points)

Skills Developed

- Serverless development

In this assignment, you will write a serverless function to fetch questions from any third-party source(s) and populate the question repository (refer Assignment 1 & 2).

Guidelines and Assumptions

- The third-party source could be like Leetcode.
- The question population can be real-time or periodic batch update.

Required Elements in Assignment 6 and Grading Rubrics

Requirement	Points
Demonstrate the serverless function you developed	1
Demonstrate the question repository being populated	1

Submission instructions

- Refer to Appendix 2 to create your team in GitHub Classroom. to create your team in GitHub Classroom.
- Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up.
- Tag the commit on the master branch that marks the end of Assignment 6 as "Assignment 6" and push the tags.
    - Ensure to include the relevant docker/docker compose files in your repository and tag the relevant commits.
- Record the demonstration that includes the following:
    - Running the serverless function to update the question repository.
    - Next, show the updated state of the question repository by opening a few new questions added by the serverless function.
    - Label the .mp4 file <your project group number>_Assignment-6.mp4 (e.g., G04_Assignment-6.mp4)
    - Time limit for the video: 2-3 minutes.
    - Submit the video to Canvas >Assignments -> Assignment-6.

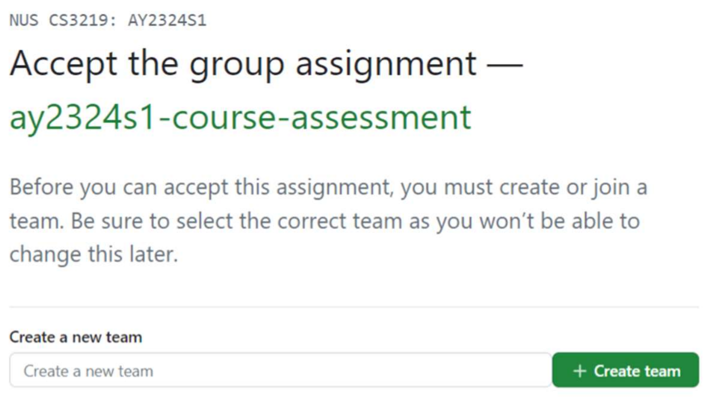
Appendix 1: Setting up of team on GitHub Classroom

We will be using GitHub Classrooms to monitor and track your assignment progress. You are required to evolve the same repository as your project repository.

You are expected to use the template repository provided and make commits to the repository during the development phase, as we may require viewing the history of the repository in case of any disputes or disagreements.

Please follow these instructions to set up your GitHub repository using GitHub Classrooms. **Please note that GitHub classroom access will be open on Thursday, 31 Aug, 1000hrs.** Before that, you may go to the link, however, you won't be able to accept the assignment.

1. Select ONE representative to click on the GitHub Classroom invitation link: https://classroom.github.com/a/6BOvYMwN
2. This representative should then be directed to this page



NUS CS3219: AY2324S1

Accept the group assignment — ay2324s1-course-assessment

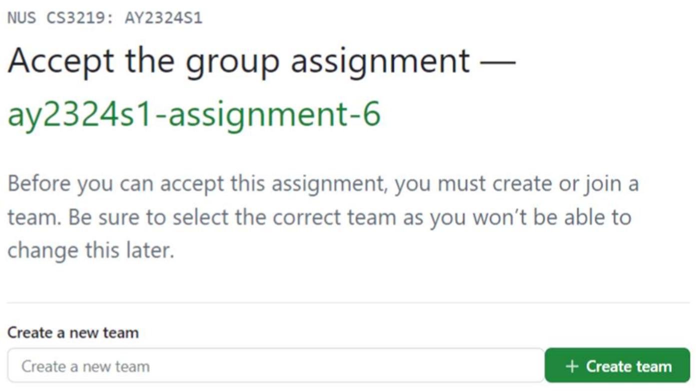
Before you can accept this assignment, you must create or join a team. Be sure to select the correct team as you won't be able to change this later.

Create a new team

3. Your team representative must now create your new team: G*<Team Number> (E.g., G01),* based on the team number from Canvas
4. Click "+ Create team"
5. Once you click the button, your team will be successfully created. Then, GitHub will ask permission for your team to access the team's GitHub repository. The team's GitHub repository name will be **`ay2324s1-course-assessment-g<team number>.`** Please click "Accept this assignment" button.
6. You will be redirected to the "Creating your repository" page. Wait for a few moments while the repository is created.
7. Once ready, you will be redirected to a "You're ready to go" page. Otherwise, you can refresh the page manually.
8. All other team members should now access the GitHub classroom link: https://classroom.github.com/a/6BOvYMwN
9. Please select your team *G<Team Number>* under "Join an existing team" and click "Join".
10. You will be directed to a "You're ready to go" confirmation page that you have joined the team.
11. Click on the link to go to the assignment repository, and you will be directed to the repository.

Appendix 2: Setting up for Assignment 6

- The process/instructions for accepting the assignment remain the same as Appendix 1.
- However, for this assignment, we will use a new GitHub classroom assignment found here: https://classroom.github.com/a/UxpU_KWG



NUS CS3219: AY2324S1

Accept the group assignment — ay2324s1-assignment-6

Before you can accept this assignment, you must create or join a team. Be sure to select the correct team as you won't be able to change this later.

- Once you click the button, your team will be successfully created. Then, GitHub will ask permission for your team to access the team's GitHub repository. The team's GitHub repository name will be **`ay2324s1-assessment-6-g<team number>.`** Please click "Accept this assignment" button.

All other team members should now access the GitHub classroom link: https://classroom.github.com/a/UxpU_KWG

Appendix 3: Sample data for Assignments 1 & 2

This section contains 2 tables. They are:

1. Table 1: Contains question ID, title, categories, complexity, and link to question.
2. Table 2: Contains question ID and corresponding question description.

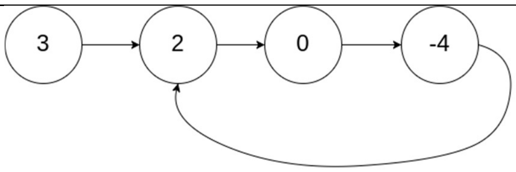
Table 1: Question ID, Title, Categories, Complexity and Links to the Questions

Question Id	**Question Title**	**Question Categories**	**Question Complexity**	**Link**
1	Reverse a String	Strings, Algorithms	Easy	https://leetcode.com/problems/reverse-string/
2	Linked List Cycle Detection	Data Structures, Algorithms	Easy	https://leetcode.com/problems/linked-list-cycle/
3	Roman to Integer	Algorithms	Easy	https://leetcode.com/problems/roman-to-integer/
4	Add Binary	Bit Manipulation, Algorithms	Easy	https://leetcode.com/problems/add-binary/
5	Fibonacci Number	Recursion, Algorithms	Easy	https://leetcode.com/problems/fibonacci-number/
6	Implement Stack using Queues	Data Structures	Easy	https://leetcode.com/problems/implement-stack-using-queues/
7	Combine Two Tables	Databases	Easy	https://leetcode.com/problems/combine-two-tables/
8	Repeated DNA Sequences	Algorithms, Bit Manipulation	Medium	https://leetcode.com/problems/repeated-dna-sequences/
9	Course Schedule	Data Structures, Algorithms	Medium	https://leetcode.com/problems/course-schedule/
10	LRU Cache Design	Data Structures	Medium	https://leetcode.com/problems/lru-cache/
11	Longest Common Subsequence	Strings, Algorithms	Medium	https://leetcode.com/problems/longest-common-subsequence/
12	Rotate Image	Arrays, Algorithms	Medium	https://leetcode.com/problems/rotate-image/
13	Airplane Seat Assignment Probability	Brainteaser	Medium	https://leetcode.com/problems/airplane-seat-assignment-probability/
14	Validate Binary Search Tree	Data Structures, Algorithms	Medium	https://leetcode.com/problems/validate-binary-search-tree/

15	Sliding Window Maximum	Arrays, Algorithms	Hard	https://leetcode.com/problems/sliding-window-maximum/
16	N-Queen Problem	Algorithms	Hard	https://leetcode.com/problems/n-queens/
17	Serialize and Deserialize a Binary Tree	Data Structures, Algorithms	Hard	https://leetcode.com/problems/serialize-and-deserialize-binary-tree/
18	Wildcard Matching	Strings, Algorithms	Hard	https://leetcode.com/problems/wildcard-matching/
19	Chalkboard XOR Game	Brainteaser	Hard	https://leetcode.com/problems/chalkboard-xor-game/
20	Trips and Users	Databases	Hard	https://leetcode.com/problems/trips-and-users/

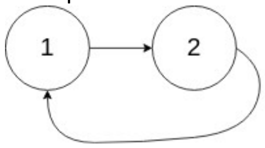
Table 2: Question ID and corresponding Question Descriptions

Question Id	Question Description
1	Write a function that reverses a string. The input string is given as an array of characters s. You must do this by modifying the input array in-place with O(1) extra memory. Example 1: Input: s = ["h","e","l","l","o"] Output: ["o","l","l","e","h"] Example 2: Input: s = ["H","a","n","n","a","h"] Output: ["h","a","n","n","a","H"] Constraints: • 1 <= s.length <= 10^5 • s[i] is a printable ascii character.
2	Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter. Return true if there is a cycle in the linked list. Otherwise, return false. Example 1:



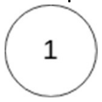
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

Example 2:



Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.

Example 3:



Input: head = [1], pos = -1
Output: false
Explanation: There is no cycle in the linked list.

Constraints:

- The number of the nodes in the list is in the range [0, 104].
- -10^5 <= Node.val <= 10^5
- pos is -1 or a valid index in the linked-list.

Follow up: Can you solve it using O(1) (i.e. constant) memory?

3

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written

	as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used: I can be placed before V (5) and X (10) to make 4 and 9. X can be placed before L (50) and C (100) to make 40 and 90. C can be placed before D (500) and M (1000) to make 400 and 900. Given a roman numeral, convert it to an integer. Example 1: Input: s = "III" Output: 3 Explanation: III = 3. Example 2: Input: s = "LVIII" Output: 58 Explanation: L = 50, V= 5, III = 3. Example 3: Input: s = "MCMXCIV" Output: 1994 Explanation: M = 1000, CM = 900, XC = 90 and IV = 4. Constraints: • 1 <= s.length <= 15 • s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M'). • It is guaranteed that s is a valid roman numeral in the range [1, 3999].
4	Given two binary strings a and b, return their sum as a binary string. Example 1: Input: a = "11", b = "1" Output: "100" Example 2: Input: a = "1010", b = "1011" Output: "10101" Constraints: • 1 <= a.length, b.length <= 104 • a and b consist only of '0' or '1' characters. • Each string does not contain leading zeros except for the zero itself.

5	The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is, F(0) = 0, F(1) = 1 F(n) = F(n - 1) + F(n - 2), for n > 1. Given n, calculate F(n). Example 1: Input: n = 2 Output: 1 Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1. Example 2: Input: n = 3 Output: 2 Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2. Example 3: Input: n = 4 Output: 3 Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3. Constraints: • 0 <= n <= 30
6	Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty). Implement the MyStack class: void push(int x) Pushes element x to the top of the stack. int pop() Removes the element on the top of the stack and returns it. int top() Returns the element on the top of the stack. boolean empty() Returns true if the stack is empty, false otherwise. Notes: You must use only standard operations of a queue, which means that only push to back, peek/pop from front, size and is empty operations are valid. Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations. Example 1: Input ["MyStack", "push", "push", "top", "pop", "empty"]

```
[[], [1], [2], [], [], []]
Output
[null, null, null, 2, 2, false]

Explanation
MyStack myStack = new MyStack();
myStack.push(1);
myStack.push(2);
myStack.top(); // return 2
myStack.pop(); // return 2
myStack.empty(); // return False
```

Constraints:

- 1 <= x <= 9
- At most 100 calls will be made to push, pop, top, and empty.
- All the calls to pop and top are valid.

Follow-up: Can you implement the stack using only one queue?

7

SQL Schema
Table: Person

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| personId    | int     |
| lastName    | varchar |
| firstName   | varchar |
+-------------+---------+
```

personId is the primary key (column with unique values) for this table.
This table contains information about the ID of some persons and their first and last names.

Table: Address

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| addressId   | int     |
| personId    | int     |
| city        | varchar |
| state       | varchar |
+-------------+---------+
```

addressId is the primary key (column with unique values) for this table.
Each row of this table contains information about the city and state of one person with ID = PersonId.

Write a solution to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:
Person table:
```
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
+----------+----------+-----------+
```
Address table:
```
+-----------+----------+---------------+------------+
| addressId | personId | city          | state      |
+-----------+----------+---------------+------------+
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
+-----------+----------+---------------+------------+
```
Output:
```
+-----------+----------+---------------+----------+
| firstName | lastName | city          | state    |
+-----------+----------+---------------+----------+
| Allen     | Wang     | Null          | Null     |
| Bob       | Alice    | New York City | New York |
+-----------+----------+---------------+----------+
```
Explanation:
There is no address in the address table for the personId = 1 so we return null in their city and state.
addressId = 1 contains information about the address of personId = 2.

8

The DNA sequence is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

For example, "ACGAATTCCG" is a DNA sequence.
When studying DNA, it is useful to identify repeated sequences within the DNA.

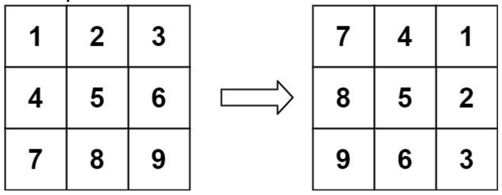
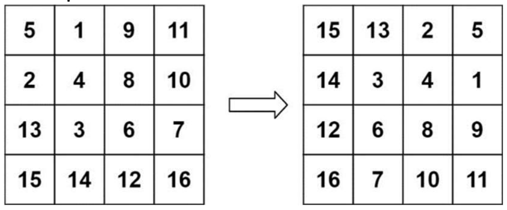
Given a string s that represents a DNA sequence, return all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in any order.

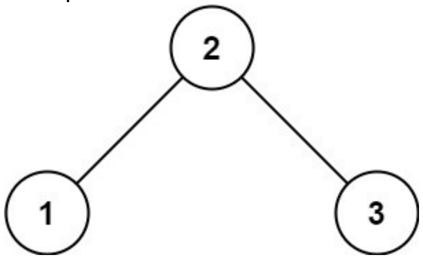
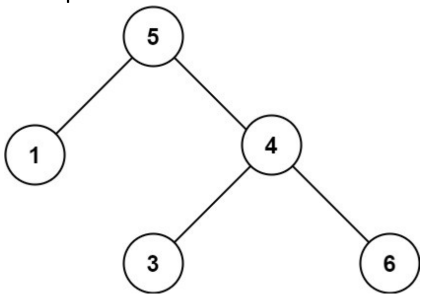
Example 1:

	Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT" Output: ["AAAAACCCCC","CCCCCAAAAA"] Example 2: Input: s = "AAAAAAAAAAAAA" Output: ["AAAAAAAAAA"] Constraints: • 1 <= s.length <= 10^5 • s[i] is either 'A', 'C', 'G', or 'T'.
9	There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai. For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1. Return true if you can finish all courses. Otherwise, return false. Example 1: Input: numCourses = 2, prerequisites = [[1,0]] Output: true Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So it is possible. Example 2: Input: numCourses = 2, prerequisites = [[1,0],[0,1]] Output: false Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So it is impossible. Constraints: • 1 <= numCourses <= 2000 • 0 <= prerequisites.length <= 5000 • prerequisites[i].length == 2 • 0 <= ai, bi < numCourses • All the pairs prerequisites[i] are unique.
10	Design a data structure that follows the constraints of a Least Recently Used (LRU) cache. Implement the LRUCache class: LRUCache(int capacity) Initialize the LRU cache with positive size capacity. int get(int key) Return the value of the key if the key exists, otherwise return -1.

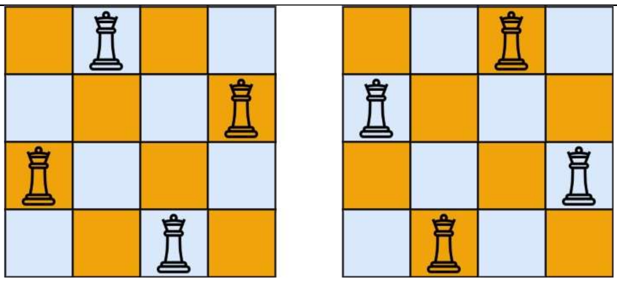
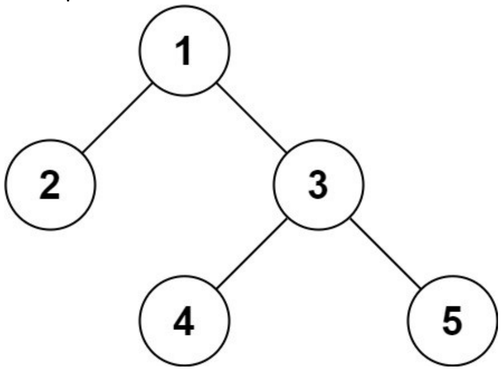
	void put(int key, int value) Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, evict the least recently used key. The functions get and put must each run in O(1) average time complexity.
	
	Example 1:
	
	Input
	["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]
	[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]
	Output
	[null, null, null, 1, null, -1, null, -1, 3, 4]
	
	Explanation
	LRUCache lRUCache = new LRUCache(2);
	lRUCache.put(1, 1); // cache is {1=1}
	lRUCache.put(2, 2); // cache is {1=1, 2=2}
	lRUCache.get(1); // return 1
	lRUCache.put(3, 3); // LRU key was 2, evicts key 2, cache is {1=1, 3=3}
	lRUCache.get(2); // returns -1 (not found)
	lRUCache.put(4, 4); // LRU key was 1, evicts key 1, cache is {4=4, 3=3}
	lRUCache.get(1); // return -1 (not found)
	lRUCache.get(3); // return 3
	lRUCache.get(4); // return 4
	
	Constraints:
	• 1 <= capacity <= 3000
	• 0 <= key <= 10^4
	• 0 <= value <= 10^5
	• At most 2 * 10^5 calls will be made to get and put.

11	Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0. A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters. For example, "ace" is a subsequence of "abcde". A common subsequence of two strings is a subsequence that is common to both strings. Example 1: Input: text1 = "abcde", text2 = "ace" Output: 3 Explanation: The longest common subsequence is "ace" and its length is 3. Example 2:

	Input: text1 = "abc", text2 = "abc"
	Output: 3
	Explanation: The longest common subsequence is "abc" and its length is 3.
	Example 3:
	Input: text1 = "abc", text2 = "def"
	Output: 0
	Explanation: There is no such common subsequence, so the result is 0.
	Constraints:
	• 1 <= text1.length, text2.length <= 1000
	• text1 and text2 consist of only lowercase English characters.
12	You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise).
	You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.
	Example 1:
	
	Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
	Output: [[7,4,1],[8,5,2],[9,6,3]]
	Example 2:
	
	Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]
	Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]
	Constraints:
	• n == matrix.length == matrix[i].length
	• 1 <= n <= 20
	• -1000 <= matrix[i][j] <= 1000
13	n passengers board an airplane with exactly n seats. The first passenger has lost the ticket and picks a seat randomly. But after that, the rest of the passengers will:
	Take their own seat if it is still available, and
	Pick other seats randomly when they find their seat occupied
	Return the probability that the nth person gets his own seat.

	Example 1: Input: n = 1 Output: 1.00000 Explanation: The first person can only get the first seat. Example 2: Input: n = 2 Output: 0.50000 Explanation: The second person has a probability of 0.5 to get the second seat (when first person gets the first seat). Constraints: • 1 <= n <= 10^5
14	Given the root of a binary tree, determine if it is a valid binary search tree (BST). A valid BST is defined as follows: • The left subtree of a node contains only nodes with keys less than the node's key. • The right subtree of a node contains only nodes with keys greater than the node's key. • Both the left and right subtrees must also be binary search trees. Example 1:  Input: root = [2,1,3] Output: true Example 2:  Input: root = [5,1,4,null,null,3,6] Output: false Explanation: The root node's value is 5 but its right child's value is 4.

	Constraints: • The number of nodes in the tree is in the range [1, 10^4]. • -2^{31} <= Node.val <= 2^{31} - 1
15	You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position. Return the max sliding window. Example 1: Input: nums = [1,3,-1,-3,5,3,6,7], k = 3 Output: [3,3,5,5,6,7] Explanation: Window position Max --------------- ----- [1 3 -1] -3 5 3 6 7 3 1 [3 -1 -3] 5 3 6 7 3 1 3 [-1 -3 5] 3 6 7 5 1 3 -1 [-3 5 3] 6 7 5 1 3 -1 -3 [5 3 6] 7 6 1 3 -1 -3 5 [3 6 7] 7 Example 2: Input: nums = [1], k = 1 Output: [1] Constraints: • 1 <= nums.length <= 10^5 • -10^4 <= nums[i] <= 10^4 • 1 <= k <= nums.length
16	The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order. Each solution contains a distinct board configuration of the n-queens' placement, where 'Q' and '.' both indicate a queen and an empty space, respectively. Example 1:

	 Input: n = 4 Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]] Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above Example 2: Input: n = 1 Output: [["Q"]] Constraints: • 1 <= n <= 9
17	Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment. Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure. Clarification: The input/output format is the same as how LeetCode serializes a binary tree. You do not necessarily need to follow this format, so please be creative and come up with different approaches yourself. Example 1:  Input: root = [1,2,3,null,null,4,5] Output: [1,2,3,null,null,4,5] Example 2: Input: root = [] Output: []

	Constraints: • The number of nodes in the tree is in the range [0, 10^4]. • -1000 <= Node.val <= 1000
18	Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where: • '?' Matches any single character. • '*' Matches any sequence of characters (including the empty sequence). The matching should cover the entire input string (not partial). Example 1: Input: s = "aa", p = "a" Output: false Explanation: "a" does not match the entire string "aa". Example 2: Input: s = "aa", p = "*" Output: true Explanation: '*' matches any sequence. Example 3: Input: s = "cb", p = "?a" Output: false Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'. Constraints: • 0 <= s.length, p.length <= 2000 • s contains only lowercase English letters. • p contains only lowercase English letters, '?' or '*'.
19	You are given an array of integers nums represents the numbers written on a chalkboard. Alice and Bob take turns erasing exactly one number from the chalkboard, with Alice starting first. If erasing a number causes the bitwise XOR of all the elements of the chalkboard to become 0, then that player loses. The bitwise XOR of one element is that element itself, and the bitwise XOR of no elements is 0. Also, if any player starts their turn with the bitwise XOR of all the elements of the chalkboard equal to 0, then that player wins. Return true if and only if Alice wins the game, assuming both players play optimally. Example 1: Input: nums = [1,1,2] Output: false Explanation: Alice has two choices: erase 1 or erase 2.

If she erases 1, the nums array becomes [1, 2]. The bitwise XOR of all the elements of the chalkboard is 1 XOR 2 = 3. Now Bob can remove any element he wants, because Alice will be the one to erase the last element and she will lose.
If Alice erases 2 first, now nums become [1, 1]. The bitwise XOR of all the elements of the chalkboard is 1 XOR 1 = 0. Alice will lose.

Example 2:
Input: nums = [0,1]
Output: true

Example 3:
Input: nums = [1,2,3]
Output: true

Constraints:
- 1 <= nums.length <= 1000
- 0 <= nums[i] < 2^{16}

20

SQL Schema
Pandas Schema

Table: Trips
```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| id          | int      |
| client_id   | int      |
| driver_id   | int      |
| city_id     | int      |
| status      | enum     |
| request_at  | date     |
+-------------+----------+
```
id is the primary key (column with unique values) for this table.
The table holds all taxi trips. Each trip has a unique id, while client_id and driver_id are foreign keys to the users_id at the Users table.
Status is an ENUM (category) type of ('completed', 'cancelled_by_driver', 'cancelled_by_client').

Table: Users

```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| users_id    | int      |
| banned      | enum     |
| role        | enum     |
+-------------+----------+
```
users_id is the primary key (column with unique values) for this table.
The table holds all users. Each user has a unique users_id, and role is an ENUM type of ('client', 'driver', 'partner').

banned is an ENUM (category) type of ('Yes', 'No').

The cancellation rate is computed by dividing the number of canceled (by client or driver) requests with unbanned users by the total number of requests with unbanned users on that day.

Write a solution to find the cancellation rate of requests with unbanned users (both client and driver must not be banned) each day between "2013-10-01" and "2013-10-03". Round Cancellation Rate to two decimal points.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:
Trips table:
```
+----+-----------+-----------+---------+---------------------+------------+
| id | client_id | driver_id | city_id | status              | request_at |
+----+-----------+-----------+---------+---------------------+------------+
| 1  | 1         | 10        | 1       | completed           | 2013-10-01 |
| 2  | 2         | 11        | 1       | cancelled_by_driver | 2013-10-01 |
| 3  | 3         | 12        | 6       | completed           | 2013-10-01 |
| 4  | 4         | 13        | 6       | cancelled_by_client | 2013-10-01 |
| 5  | 1         | 10        | 1       | completed           | 2013-10-02 |
| 6  | 2         | 11        | 6       | completed           | 2013-10-02 |
| 7  | 3         | 12        | 6       | completed           | 2013-10-02 |
| 8  | 2         | 12        | 12      | completed           | 2013-10-03 |
| 9  | 3         | 10        | 12      | completed           | 2013-10-03 |
| 10 | 4         | 13        | 12      | cancelled_by_driver | 2013-10-03 |
+----+-----------+-----------+---------+---------------------+------------+
```
Users table:
```
+----------+--------+--------+
| users_id | banned | role   |
+----------+--------+--------+
| 1        | No     | client |
| 2        | Yes    | client |
| 3        | No     | client |
| 4        | No     | client |
| 10       | No     | driver |
| 11       | No     | driver |
| 12       | No     | driver |
| 13       | No     | driver |
+----------+--------+--------+
```
Output:
```
+------------+-------------------+
| Day        | Cancellation Rate |
```

	+------------+-------------------+ | 2013-10-01 | 0.33 | | 2013-10-02 | 0.00 | | 2013-10-03 | 0.50 | +------------+-------------------+ Explanation: On 2013-10-01: - There were 4 requests in total, 2 of which were canceled. - However, the request with Id=2 was made by a banned client (User_Id=2), so it is ignored in the calculation. - Hence there are 3 unbanned requests in total, 1 of which was canceled. - The Cancellation Rate is (1 / 3) = 0.33 On 2013-10-02: - There were 3 requests in total, 0 of which were canceled. - The request with Id=6 was made by a banned client, so it is ignored. - Hence there are 2 unbanned requests in total, 0 of which were canceled. - The Cancellation Rate is (0 / 2) = 0.00 On 2013-10-03: - There were 3 requests in total, 1 of which was canceled. - The request with Id=8 was made by a banned client, so it is ignored. - Hence there are 2 unbanned request in total, 1 of which were canceled. - The Cancellation Rate is (1 / 2) = 0.50