Softwarica College of IT & E-Commerce
STW5008CEM: Programming For Developers

in collaboration with

Assignment Brief 2024

Module Name STW5008CEM: Programming for Developers	Ind/Group **Individual**	Cohort **March 2024 - Regular**	Module Code: STW5008CEM
Coursework Title (e.g., CWK)			Hand out date: 07/01/2024
Lecturer: Hikmat Saud			Due date: 08/18/2024
Estimated Time (hrs): Word Limit*: n/a	Coursework type: Individual / Practical		% of Module Mark 33%
Submission arrangement online via Softwarica Schools works Pro: Upload through Assignment links File types and method of recording: URLs (source code repositories) Mark and Feedback date: Within 3 weeks of assignment submission Mark and Feedback method: Rubric marks and comments via Softwarica LMS			

Module Learning Outcomes Assessed:

- Understand and select appropriate algorithms for solving a range of problems and reason for their complexity and efficiency.
- Design and implement algorithms and data structures for novel problems.
- Understand the intractability of certain problems and implement approaches to estimate the solution to intractable problems.
- Describe the issue of data consistency in non-synchronous applications. Design and implement a basic concurrent application.

Notes:

1. You are expected to use the CUHarvard referencing format. For support and advice on how this students can contact Centre for Academic Writing (CAW).
2. Please notify your registry course support team and module leader for disability support.
3. The University cannot take responsibility for any coursework lost or corrupted on disks, laptops or personal computer. Students should therefore regularly back-up any work and are advised to save it on the University system.
4. If there are technical or performance issues that prevent students submitting coursework through the online coursework submission system on the day of a coursework deadline, an appropriate extension to the coursework submission deadline will be agreed. This extension will normally be 24 hours or the next working day if the deadline falls on a Friday or over the weekend period. This will
be communicated via email and as a Softwarica Moodle announcement.

Question1

a)

Imagine you're a scheduling officer at a university with n classrooms numbered 0 to n-1. Several different courses require classrooms throughout the day, represented by an array of classes classes[i] = [starti, endi], where starti is the start time of the class and endi is the end time (both in whole hours). Your goal is to assign each course to a classroom while minimizing disruption and maximizing classroom utilization. Here are the rules:

- Priority Scheduling: Classes with earlier start times have priority when assigning classrooms. If multiple classes start at the same time, prioritize the larger class (more students).
- Dynamic Allocation: If no classroom is available at a class's start time, delay the class until a room becomes free. The delayed class retains its original duration.
- Room Release: When a class finishes in a room, that room becomes available for the next class with the highest priority (considering start time and size).

Your task is to determine which classroom held the most classes throughout the day. If multiple classrooms are tied, return the one with the lowest number.

Example:

Example 1:

Input: n = 2, classes = [[0, 10], [1, 5], [2, 7], [3, 4]]

Output: 0

Explanation:

- At time 0, both classes are not being used. The first class starts in room 0.

- At time 1, only room 1 is not being used. The second class starts in room 1.

- At time 2, both rooms are being used. The third class is delayed.

- At time 3, both rooms are being used. The fourth class is delayed.

- At time 5, the class in room 1 finishes. The third class starts in room 1 for the time period [5, 10).

- At time 10, the classes in both rooms finish. The fourth class starts in room 0 for the time period [10,11).

Both rooms 0 and 1 held 2 classes, so we return 0.

Example 2:

Input: n = 3, meetings = [[1, 20],[2,10],[3,5],[4,9],[6,8]]

Output: 1

Explanation:

- At time 1, all three rooms are not being used. The first class starts in room 0.

- At time 2, rooms 1 and 2 are not being used. The second class starts in room 1.

- At time 3, only room 2 is not being used. The third class starts in room 2.

- At time 4, all three rooms are being used. The fourth class is delayed.

- At time 5, the class in room 2 finishes. The fourth class starts in room 2 for the time period [5, 10).

- At time 6, all three rooms are being used. The fifth class is delayed.

- At time 10, the class in rooms 1 and 2 finish. The fifth class starts in room 1 for the time period [10, 12).

Room 0 held 1 class while rooms 1 and 2 each held 2 classes, so we return 1.

[5 Marks]

b)

Imagine you have a secret decoder ring with rotating discs labeled with the lowercase alphabet. You're given a message s written in lowercase letters and a set of instructions shifts encoded as tuples (start_disc, end_disc, direction). Each instruction represents rotating the discs between positions start_disc and end_disc (inclusive) either clockwise (direction = 1) or counter-clockwise (direction = 0). Rotating a disc shifts the message by one letter for each position moved on the alphabet (wrapping around from 'z' to 'a' and vice versa).

Your task is to decipher the message s by applying the rotations specified in shifts in the correct order.

Example:

In Input: s = "hello", shifts = [[0, 1, 1], [2, 3, 0], [0, 2, 1]]

Output: jglko

Shifts:

(0, 1, 1) - Rotate discs 0 and 1 clockwise (h becomes i, e becomes f)

(2, 3, 0) - Rotate discs 2 and 3 counter-clockwise (both l becomes k)

(0, 2, 1) - Rotate discs 0, 1, and 2 clockwise (i becomes j, f becomes g, and k becomes l)

[5 Marks]

Question2

a)

You are tasked with implementing a basic calculator with a graphical user interface (GUI) in Java. The calculator should be able to evaluate valid mathematical expressions entered by the user and display the result on the GUI.

Specifications:

1. Create a Java GUI application named "BasicCalculatorGUI" that extends JFrame.
2. The GUI should contain the following components:

- A JTextField for the user to input the mathematical expression.
- A JButton labeled "Calculate" that triggers the evaluation of the expression.
- A JLabel to display the result of the calculation.

3. When the "Calculate" button is clicked, the application should:

- Retrieve the expression entered by the user from the text field.
- Evaluate the expression using a custom algorithm (without using any built-in function that evaluates strings as mathematical expressions, such as eval()).
- Display the result of the evaluation on the label.

4. The calculator should support basic arithmetic operations, including addition, subtraction, multiplication, and division. It should also handle parentheses for grouping expressions.
5. The GUI should have an appropriate layout and size to ensure usability and aesthetics.
6. Ensure proper error handling and validation of user input. Display error messages if the input expression is invalid or cannot be evaluated.

   Example 1:

   Input: s = "1 + 1"

   Output: 2

   Example 2:

   Input: s = " 2-1 + 2 "

   Output: 3

   Example 3:

   Input: s = "(1+(4+5+2)-3)+(6+8)"

   Output: 23

[5 Marks]

b)

Imagine you're at a movie theater with assigned seating. You have a seating chart represented by an array nums where nums[i] represents the seat number at row i. You're looking for two friends to sit together, considering their seat preferences and your limitations:

Seating Distance: Your friends prefer to sit close together, with a maximum allowed seat difference of indexDiff. Imagine indexDiff = 2, meaning they'd be comfortable within 2 seats of each other (e.g., seats 3 and 5).

Movie Preference: They also want similar movie tastes, requiring a difference in their seat numbers (abs(nums[i] - nums[j])) to be within valueDiff. Think of valueDiff = 1 as preferring movies with similar ratings (e.g., seats 4 and 5 for movies rated 4.5 and 5 stars).

Your task is to determine if there are two friends (represented by two indices i and j) who can sit together while satisfying both the seating distance and movie preference limitations.

Example:

Input:

Seating chart: nums = [2, 3, 5, 4, 9]

Seating distance limit: indexDiff = 2

Movie preference limit: valueDiff = 1

Output: true

Possible pairs:

(0, 1): Seat difference (1) within indexDiff, movie difference (1) within valueDiff.

(3, 4): Seat difference (1) within indexDiff, movie difference (5) exceeds valueDiff.

In this scenario, you can find two friends who can sit together, so the answer would be true.

[5 Marks]

Question 3

a)

Imagine a small community with n houses, numbered 0 to n-1. Some houses have restrictions against becoming friends, represented by pairs in the restrictions list. For example, if [0, 1] is in the list, houses 0 and 1 cannot be directly or indirectly friends (through common friends).

Residents send friend requests to each other, represented by pairs in the requests list. Your task is to determine if each friend request can be accepted based on the current friendship network and the existing restrictions.

Example2:

Input:

Number of houses: 3

Restrictions: [0, 1]

[0, 1] (House 0 and House 1 cannot be friends)

Friend requests: [[0, 2], [2,1]]

[0, 2] (House 0 requests friendship with House 2)

[2, 1] (House 2 requests friendship with House 1)

Outcome: [approved, denied]

Request 0: Approved (House 0 and 2 don't have any restrictions)

Request 1: Denied (House 2 and 1 would be indirectly connected through House 0, violating the restriction).

Example 2:

Input:

Number of Houses = 5

Restrictions = [[0, 1], [1, 2], [2, 3]]

Requests = [[0, 4], [1, 2], [3, 1], [3, 4]]

Output: [approved, denied, approved, denied]

Explanation:

Request 0: house 0 and house 4 can be friends, so they become direct friends.

Request 1: house 1 and house 2 cannot be friends since they are directly restricted.

Request 2: house 3 and house 1 can be friends, so they become direct friends.

Request 3: house 3 and house 4 cannot be friends since person 0 and person 1 would be indirect friends (0--4--3--1).

[5 Marks]

b)

Imagine you are managing a bus service with passengers boarding at various stops along the route. Your task is to optimize the boarding process by rearranging the order of passengers at intervals of k stops, where k is a positive integer. If the total number of passengers is not a multiple of k, then the remaining passengers at the end of the route should maintain their original order.

Example1:

Suppose the boarding sequence of passengers is Input: head = [1, 2, 3, 4, and 5], k = 2

Output: [2, 1, 4, 3, and 5] after optimizing the boarding process

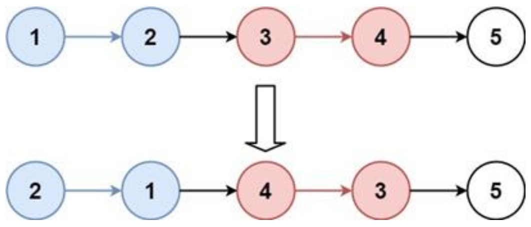
Explanation:

Passengers at positions 1 to 2 ([1, 2]) are rearranged as the given interval is k=2. Therefore, the first interval includes two initial passengers that are rearranged to [2, 1].

Another multiple of interval k is 4 (2*2), therefore passengers at positions 3 to 4 are rearranged.

Remaining passengers ([5]) stay unchanged as their position is 5 and does not fall within a multiple of the given interval k=2 i.e. 2*3= 6 to rearrange remaining element there must exist element at position 5 and 6.

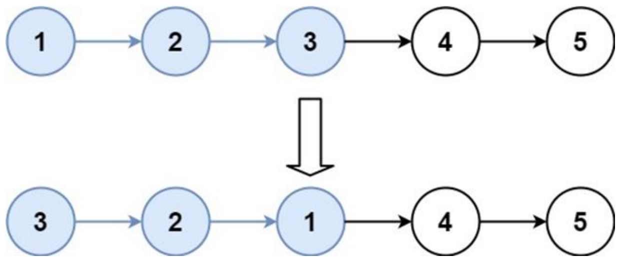
So, the modified boarding sequence becomes [2, 1, 4, 3, and 5].



Example2:

Input: head = [1,2,3,4,5], k = 3

Output: [3,2,1,4,5]



[5 Marks]

Question 4

a)

Imagine you're a city planner tasked with improving the road network between key locations (nodes) represented by numbers 0 to n-1. Some roads (edges) have known travel times (positive weights), while others are under construction (weight -1). Your goal is to modify the construction times on these unbuilt roads to achieve a specific travel time (target) between two important locations (from source to destination).

Constraints:

You can only modify the travel times of roads under construction (-1 weight).

The modified times must be positive integers within a specific range. [1, 2 * 10^9]

You need to find any valid modification that achieves the target travel time.

Examples:

Input:

City: 5 locations

Roads: [[4, 1,-1], [2, 0,-1],[0,3,-1],[4,3,-1]]

Source: 0, Destination: 1, Target time: 5 minutes

Output: [[4,1,1],[2,0,1],[0,3,3],[4,3,1]]

Solution after possible modification

[5 Marks]

b)

Imagine you're on a treasure hunt in an enchanted forest represented by a binary tree root. Each node in the tree has a value representing a magical coin. Your goal is to find the largest collection of coins that forms a magical grove.

A magical grove is defined as a subtree where:

Every coin's value on left subtree less than to the value of the coin directly above it (parent node).

Every coin's value on right subtree is greater than to the value of the coin directly above it (parent node).

Every coin in the grove needs to be binary search tree.

Your task is to find the magical grove with the highest total value of coins.

Examples:

Forest: [1,4,3,2,4,2,5,null,null,null,null,null,null,4,6]

Output: 20

Largest Magical Grove: The subtree rooted at node 3, with a total value of 20 (3+2+5+4+6).

[5 Marks]

Question 5

a) Implement travelling a salesman problem using hill climbing algorithm.

[5 Marks]

b)

Imagine you're on a challenging hiking trail represented by an array nums, where each element represents the altitude at a specific point on the trail. You want to find the longest consecutive stretch of the trail you can hike while staying within a certain elevation gain limit (at most k).

Constraints:

You can only go uphill (increasing altitude).

The maximum allowed elevation gain between two consecutive points is k.

Goal: Find the longest continuous hike you can take while respecting the elevation gain limit.

Examples:

Input:

Trail: [4, 2, 1, 4, 3, 4, 5, 8, and 15], Elevation gain limit (K): 3

Output: 5

Explanation

Longest hike: [1, 3, 4, 5, 8] (length 5) - You can climb from 1 to 3 (gain 2), then to 4 (gain 1), and so on, all within the limit.

Invalid hike: [1, 3, 4, 5, 8, 15] - The gain from 8 to 15 (7) exceeds the limit.

[5 Marks]

Question 6

This scenario presents another sample Java GUI application using multithreading and an asynchronous framework (SwingWorker) to demonstrate asynchronous progress updates and batch processing.

Features:

File selection dialog: Choose multiple files for conversion.

Conversion options: Select the desired output format (e.g., PDF to Docx, image resize).

Start button: Initiates conversion of selected files concurrently.

Progress bar: Shows overall conversion progress with individual file indicators.

Status bar: Displays information about each file being processed (name, conversion type, progress).

Cancel button: Allows stopping the entire conversion process or individual files.

Completion notification: Provides a message when all conversions are finished.

Challenges:

Efficiently manage multiple file conversions using separate threads.

Update the GUI asynchronously to show individual file progress without blocking the main thread.

Handle potential errors during file access or conversion and provide informative feedback.

Allow cancelling specific files or the entire process gracefully.

Implementation:

Swing GUI: Design a graphical interface using Swing components for file selection, buttons, progress bars, and status messages.

Multithreading: Use a thread pool to manage multiple conversion threads efficiently.

[20 Marks]

Question7

Route Optimization for Delivery Service (Java GUI)

This scenario explores a Java GUI application using graph algorithms to optimize delivery routes for a courier service.

Features:

Delivery list: Import or manually enter a list of delivery points with addresses and order priorities.

Algorithm selection: Choose an optimization algorithm for route planning.

Vehicle options: Specify vehicle capacity and driving distance constraints.

Optimize button: Initiates the chosen algorithm to calculate the optimal delivery route.

Route visualization: Highlight the calculated route, showing stops and travel distances.

Challenges:

Implement chosen optimization algorithms using multithreading for faster calculations.

Visualize the calculated route effectively with clear highlighting and labeling.

Handle vehicle constraints and capacity limitations during route planning.

Implementation:

Swing/JavaFX GUI: Design a graphical interface using appropriate libraries to:

Provide input options for delivery list, vehicle specifications, and algorithm selection.

Show the optimized route visually.

User input on delivery list, vehicle options, and algorithm selection.

Initiating the route optimization and handling completion.

Interacting with the map display (optional).

[30 Marks]

[Total 100 Marks]

Marking Notes

1. All submitted coursework will be assessed via VIVA conducted at the end of this semester.
2. Each VIVA will last 20 minutes.
3. You will submit on the deadline a document (PDF or Word) on Moodle containing all the coursework tasks solved and including a link to your GitHub Classroom repository shared via schools works pro.
4. During the VIVA you will be assessed with a few relevant random questions.
5. If you submit only some of the tasks, your mark will be proportional to that.
6. The marking criteria for each sub question from 1 to 5 is presented below

Assignment Component	Max Marks	Grading			
Algorithm Design & Implementation	3	Develops a logical and efficient approach to solve the problem (algorithm design). Chosen algorithm is fully functional without any error. (Demonstrates good clarity in understanding the problem statement). . Output is not formatted. **(3 Marks)**	Develops a logical approach to solve the problem (algorithm design). Implements the chosen algorithm is partially functional. May have minor syntax errors or logic flaws. **(2 Marks)**	Partial Completeness of algorithm with syntax and logical flaws, output is not formatted. **(1 Marks)**	Not provided any solution. **(0 Marks)**
Testing & Validation	2 Marks		Decision control logic, loop logic in program exhibits proper functional behavior and Output is obtained for varying sets of input data. **(2 Marks)**	Decision control logic, loop logic in program lacks proper functional behavior and Output is obtained for only few sets of input data. **(1 Mark)**	No test cases were implemented to verify the program's behavior with different inputs. Additionally, the program doesn't produce any expected output, hindering its functionality. **(0 Marks)**
Additional Considerations	1 Mark			Completeness of code, well-structured code, consistent variable naming and formatting, well Commented code. Code readability,	Missing comments, program lack readability and coding standards **(0 Mark)**

				clarity, and adherence to coding standards. **(1 Mark)**	

7. The marking criteria valid for question 6 and 7 is presented below.

Criteria	0	1	2	3	4	5
Feature complete (20 %)	Not submitted	Only a few features implemented and are not executing	Many of the features are implemented but are not executing correctly	Many of the features are implemented and are executed correctly	Most of the features are implemented and are executed correctly	All features implemented and are executed correctly
Code aesthetic (15%)	Not submitted	Assignment submitted but not commented and formatted. variable’s/classes/ function are defined but meaningless	Lack of comments, formatted in Source code. Only a few classes and functions are defined but hard to read	Lack of comments, formatted in Source code, but meaningful variable/class/ function names are used few functions are defined.	Lack of comments, formatted in Source code, but meaningful variable/class/ function names are used. Code is easy to read	Source code is well commented, properly formatted, meaningful variable/function/class names are used. Code is easy to read and understand, having many pure functions.
GUI (20%)	Not submitted	Hard to use. Only some components are used and unmanaged	Few frames are difficult to use. UI components are used but unmanaged.	Some frames are difficult to use. UI components are used but unmanaged.	Easy to use, Proper use of various UI components. User Interaction is low	Easy to use, Proper use of various UI components, Clean and interactive UI
I/P Validation (10%)	Not submitted	Only a few input fields are validated. Error message are not shown	Only a few input fields are validated. Error messages are shown in code format	Most input fields are properly validated. Error messages are shown in code format	Most input fields are properly validated. Error messages are properly shown in natural language	All input fields are properly validated. Error messages are properly shown in natural language.
Use of required algorithm (20%)	Not submitted	Uses an incorrect or incomplete algorithm	Only few required algorithm are well implemented	Implemented all required algorithm with few errors or missing steps.	All algorithm are implemented without any error or missing steps.	The implemented algorithms achieve optimal solutions. The program exhibits highly efficient execution of all algorithms.
Viva (15%)	Not present (Assignment submitted but absent in viva)	Could not explain the reasoning behind the code. But answered only one viva question	Could explain basic terms but not about algorithm. But answered only two viva question	Could explain reasoning behind the code, including use of loops, conditions, algorithms. answered only three viva question	Could explain reasoning behind the code, including use of loops, conditions, algorithms. answered only four viva question	Could explain reasoning behind the code, including use of loops, conditions, algorithms. Answered all five questions