Softwarica College of IT & E-Commerce
ST5008CEM: Programming For Developers

in collaboration with

Softwarica College of IT & E-commerce | Coventry University

Assignment Brief 2024

Module Name ST5008CEM: Programming for Developers	Ind/Group **Individual**	Cohort **Sept 2024** - **Regular**	Module Code: ST5008CEM
Coursework Title: Coursework			Hand out date:19 January, 2025
Lecturer: Hikmat Saud			Due date: 2 March, 2025
Estimated Time (hrs): Word Limit*: n/a	Coursework type: Individual / Practical		% of Module Mark 33%
Submission arrangement online via Schoolworkspro: Upload through Assignment links File types and method of recording: PDF document via SchoolWorksPro Mark and Feedback date: TBD Mark and Feedback method: Rubric marks and comments via Schoolworkspro			

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 Coventry University APA style for 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 Schoolworkspro announcement.

Question1
a)

You have a material with n temperature levels. You know that there exists a critical temperature f where 0 <= f <= n such that the material will react or change its properties at temperatures higher than f but remain unchanged at or below f.

Rules:

- You can measure the material's properties at any temperature level once.
- If the material reacts or changes its properties, you can no longer use it for further measurements.
- If the material remains unchanged, you can reuse it for further measurements.

Goal:

Determine the minimum number of measurements required to find the critical temperature.

Input:

- k: The number of identical samples of the material.
- n: The number of temperature levels.

Output:

- The minimum number of measurements required to find the critical temperature.

Example 1:

Input: k = 1, n = 2

Output: 2

Explanation:

Check the material at temperature 1. If its property changes, we know that f = 0.
Otherwise, raise temperature to 2 and check if property changes. If its property changes, we know that f = 1.If its property changes at temperature, then we know f = 2.
Hence, we need at minimum 2 moves to determine with certainty what the value of f is.
Example 2:
Input: k = 2, n = 6
Output: 3
Example 3:
Input: k = 3, n = 14
Output: 4

[5 Marks]

b)

You have two sorted arrays of investment returns, returns1 and returns2, and a target number k. You want to find the kth lowest combined return that can be achieved by selecting one investment from each array.

Rules:

- The arrays are sorted in ascending order.
- You can access any element in the arrays.

Goal:

Determine the kth lowest combined return that can be achieved.

Input:

- returns1: The first sorted array of investment returns.
- returns2: The second sorted array of investment returns.
- k: The target index of the lowest combined return.

Output:

- The kth lowest combined return that can be achieved.

Example 1:

Input: returns1= [2,5], returns2= [3,4], k = 2
Output: 8
Explanation: The 2 smallest investments are are:
- returns1 [0] * returns2 [0] = 2 * 3 = 6
- returns1 [0] * returns2 [1] = 2 * 4 = 8
The 2nd smallest investment is 8.
Example 2:
Input: returns1= [-4,-2,0,3], returns2= [2,4], k = 6
Output: 0
Explanation: The 6 smallest products are:
- returns1 [0] * returns2 [1] = (-4) * 4 = -16
- returns1 [0] * returns2 [0] = (-4) * 2 = -8
- returns1 [1] * returns2 [1] = (-2) * 4 = -8
- returns1 [1] * returns2 [0] = (-2) * 2 = -4
- returns1 [2] * returns2 [0] = 0 * 2 = 0
- returns1 [2] * returns2 [1] = 0 * 4 = 0
The 6th smallest investment is 0.

[5 Marks]

Question 2

a)

You have a team of n employees, and each employee is assigned a performance rating given in the integer array ratings. You want to assign rewards to these employees based on the following rules:
Every employee must receive at least one reward.
Employees with a higher rating must receive more rewards than their adjacent colleagues.
Goal:
Determine the minimum number of rewards you need to distribute to the employees.
Input:
ratings: The array of employee performance ratings.
Output:
The minimum number of rewards needed to distribute.

Example 1:

Input: ratings = [1, 0, 2]
Output: 5
Explanation: You can allocate to the first, second and third employee with 2, 1, 2 rewards respectively.

Example 2:

Input: ratings = [1, 2, 2]
Output: 4
Explanation: You can allocate to the first, second and third employee with 1, 2, 1 rewards respectively. The third employee gets 1 rewards because it satisfies the above two conditions.

[5 Marks]

b)

You have two points in a 2D plane, represented by the arrays x_coords and y_coords. The goal is to find the lexicographically pair i.e. (i, j) of points (one from each array) that are closest to each other.
Goal:
Determine the lexicographically pair of points with the smallest distance and smallest distance calculated using
| x_coords [i] - x_coords [j]| + | y_coords [i] - y_coords [j]|

Note that

|x| denotes the absolute value of x.
A pair of indices (i1, j1) is lexicographically smaller than (i2, j2) if i1 < i2 or i1 == i2 and j1 < j2.
Input:
x_coords: The array of x-coordinates of the points.
y_coords: The array of y-coordinates of the points.
Output:
The indices of the closest pair of points.
Input: x_coords = [1, 2, 3, 2, 4], y_coords = [2, 3, 1, 2, 3]
Output: [0, 3]
Explanation: Consider index 0 and index 3. The value of | x_coords [i]- x_coords [j]| + | y_coords [i]- y_coords [j]| is 1, which is the smallest value we can achieve.

[5 Marks]

Question 3

You have a network of n devices. Each device can have its own communication module installed at a cost of modules [i - 1]. Alternatively, devices can communicate with each other using direct connections. The cost of connecting two devices is given by the array connections where each connections[j] = [device1j, device2j, costj] represents the cost to connect devices device1j and device2j. Connections are bidirectional, and there could be multiple valid connections between the same two devices with different costs.

Goal:

Determine the minimum total cost to connect all devices in the network.

Input:

n: The number of devices.

modules: An array of costs to install communication modules on each device.

connections: An array of connections, where each connection is represented as a triplet [device1j, device2j, costj].

Output:

The minimum total cost to connect all devices.

Example:

Input: n = 3, modules = [1, 2, 2], connections = [[1, 2, 1], [2, 3, 1]] Output: 3

Explanation:

The best strategy is to install a communication module on the first device with cost 1 and connect the other devices to it with cost 2, resulting in a total cost of 3.

[5 Maks]

b)

A Game of Tetris

Functionality:

Queue: Use a queue to store the sequence of falling blocks.

Stack: Use a stack to represent the current state of the game board.

GUI:

A game board with grid cells.

A preview area to show the next block.

Buttons for left, right, and rotate.

Implementation:

Initialization:

- Create an empty queue to store the sequence of falling blocks.
- Create an empty stack to represent the game board.
- Initialize the game board with empty cells.
- Generate a random block and enqueue it.

Game Loop:

While the game is not over:

- Check for game over: If the top row of the game board is filled, the game is over.
- Display the game state: Draw the current state of the game board and the next block in the preview area.

Handle user input:

- If the left or right button is clicked, move the current block horizontally if possible.
- If the rotate button is clicked, rotate the current block if possible.
- Move the block: If the current block can move down without colliding, move it down. Otherwise:
- Push the current block onto the stack, representing its placement on the game board.
- Check for completed rows: If a row is filled, pop it from the stack and add a new empty row at the top.
- Generate a new random block and enqueue it.

Game Over:

- Display a game over message and the final score.

Data Structures:

Block: A class or struct to represent a Tetris block, including its shape, color, and current position.

GameBoard: A 2D array or matrix to represent the game board, where each cell can be empty or filled with a block.

Queue: A queue to store the sequence of falling blocks.

Stack: A stack to represent the current state of the game board.

Additional Considerations:

Collision detection: Implement a function to check if a block can move or rotate without colliding with other blocks or the game board boundaries.

Scoring: Implement a scoring system based on factors like completed rows, number of blocks placed, and other game-specific rules.

Leveling: Increase the speed of the falling blocks as the player's score increases.

Power-ups: Add power-ups like clearing lines, adding extra rows, or changing the shape of the current block.

[15 Marks]

Question 4

a)

Input:

Tweets table:

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| user_id     | int     |
| tweet_id    | int     |
| tweet_date  | date    |
| tweet       | varchar |
+-------------+---------+
tweet_id is the primary key (column with unique values) for this table.
Each row of this table contains user_id, tweet_id, tweet_date and tweet.
```

Write a solution to find the **top** 3 trending **hashtags** in **February** 2024. Every tweet may contain **several hashtags**.

Return *the result table ordered by count of hashtag, hashtag in* **descending** *order.*

The result format is in the following example.

Example 1:

Input:

Tweets table:

user_id	tweet_id	tweet	tweet_date
135	13	Enjoying a great start to the day. #HappyDay #MorningVibes	2024-02-01
136	14	Another #HappyDay with good vibes! #FeelGood	2024-02-03
137	15	Productivity peaks! #WorkLife #ProductiveDay	2024-02-04
138	16	Exploring new tech frontiers. #TechLife #Innovation	2024-02-04
139	17	Gratitude for today's moments. #HappyDay #Thankful	2024-02-05
140	18	Innovation drives us. #TechLife #FutureTech	2024-02-07
141	19	Connecting with nature's serenity. #Nature #Peaceful	2024-02-09

Output:

hashtag	count
#HappyDay	3
#TechLife	2
#WorkLife	1

Explanation:

#HappyDay: Appeared in tweet IDs 13, 14, and 17, with a total count of 3 mentions.
#TechLife: Appeared in tweet IDs 16 and 18, with a total count of 2 mentions.
#WorkLife: Appeared in tweet ID 15, with a total count of 1 mention.
Note: Output table is sorted in descending order by hashtag_count and hashtag respectively.

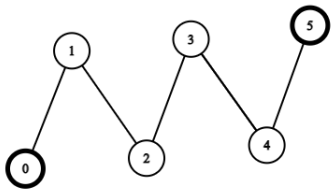
[5 Marks]

b)
You have a map of a city represented by a graph with n nodes (representing locations) and edges where edges[i] = [ai, bi] indicates a road between locations ai and bi. Each location has a value of either 0 or 1, indicating whether there is a package to be delivered. You can start at any location and perform the following actions:

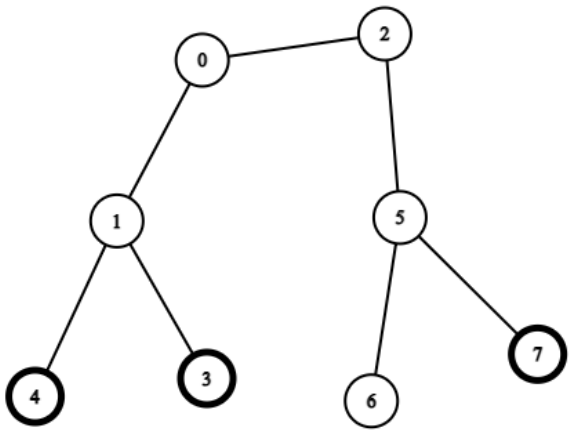
Collect packages from all locations within a distance of 2 from your current location.
Move to an adjacent location.
Your goal is to collect all packages and return to your starting location.
Goal:
Determine the minimum number of roads you need to traverse to collect all packages.
Input:
packages: An array of package values for each location.
roads: A 2D array representing the connections between locations.
Output:
The minimum number of roads to traverse.

Note that if you pass a roads several times, you need to count it into the answer several times.

Input: packages = [1, 0, 0, 0, 0, 1], roads = [[0, 1], [1, 2], [2, 3], [3, 4], [4, 5]]
Output:2



Explanation: Start at location 2, collect the packages at location 0, move to location 3, collect the packages at location 5 then move back to location 2.



Input: packages = [0,0,0,1,1,0,0,1], roads = [[0,1],[0,2],[1,3],[1,4],[2,5],[5,6],[5,7]]
Output: 2
Explanation: Start at location 0, collect the package at location 4 and 3, move to location 2, collect the package at location 7, then move back to location 0.

[5 Marks]

Question 5
Optimizing a Network with Multiple Objectives

Problem:

Suppose you are hired as software developer for certain organization and you are tasked with creating a GUI application that helps network administrators design a network topology that is both cost-effective and efficient for data transmission. The application needs to visually represent servers and clients as nodes in a graph, with potential network connections between them, each having associated costs and bandwidths. The goal is to enable the user to find a network topology that minimizes both the total cost and the latency of data transmission.

Approach:

1. **Visual Representation of the Network:**
    - Design the GUI to allow users to create and visualize a network graph where each node represents a server or client, and each edge represents a potential network connection. The edges should display associated costs and bandwidths.
2. **Interactive Optimization:**
    - Implement tools within the GUI that enable users to apply algorithms or heuristics to optimize the network. The application should provide options to find the best combination of connections that minimizes the total cost while ensuring all nodes are connected.
3. **Dynamic Path Calculation:**
    - Include a feature where the user can calculate the shortest path between any pair of nodes within the selected network topology. The GUI should display these paths, taking into account the bandwidths as weights.
4. **Real-time Evaluation:**
    - Provide real-time analysis within the GUI that displays the total cost and latency of the current network topology. If the user is not satisfied with the results, they should be able to adjust the topology and explore alternative solutions interactively.

Example:

- **Input:** The user inputs a graph in the application, representing servers, clients, potential connections, their costs, and bandwidths.
- **Output:** The application displays the optimal network topology that balances cost and latency, and shows the shortest paths between servers and clients on the GUI.

[30 Marks]

Question 6
a)

You are given a class `NumberPrinter` with three methods: `printZero`, `printEven`, and `printOdd`. These methods are designed to print the numbers 0, even numbers, and odd numbers, respectively.

Task:

Create a `ThreadController` class that coordinates three threads:

5. **ZeroThread:** Calls `printZero` to print 0s.
6. **EvenThread:** Calls `printEven` to print even numbers.
7. **OddThread:** Calls `printOdd` to print odd numbers.

These threads should work together to print the sequence "0102030405..." up to a specified number `n`. The output should be interleaved, ensuring that the numbers are printed in the correct order.

Example:

If `n = 5`, the output should be "0102030405".

Constraints:

- The threads should be synchronized to prevent race conditions and ensure correct output.
- The `NumberPrinter` class is already provided and cannot be modified.

[5 Marks]

b)
Scenario: A Multithreaded Web Crawler
Problem:
You need to crawl a large number of web pages to gather data or index content. Crawling each page sequentially can be time-consuming and inefficient.
Goal:
Create a web crawler application that can crawl multiple web pages concurrently using multithreading to improve performance.
Tasks:
Design the application:
Create a data structure to store the URLs to be crawled.
Implement a mechanism to fetch web pages asynchronously.
Design a data storage mechanism to save the crawled data.
Create a thread pool:
Use the ExecutorService class to create a thread pool for managing multiple threads.
Submit tasks:
For each URL to be crawled, create a task (e.g., a Runnable or Callable object) that fetches the web page and processes the content.
Submit these tasks to the thread pool for execution.
Handle responses:
Process the fetched web pages, extracting relevant data or indexing the content.
Handle errors or exceptions that may occur during the crawling process.
Manage the crawling queue:
Implement a mechanism to manage the queue of URLs to be crawled, such as a priority queue or a breadth-first search algorithm.

By completing these tasks, you will create a multithreaded web crawler that can efficiently crawl large numbers of web page

[15 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 Schoolworkspro 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 Consideratio	1 Mark			Completeness of code, well-	Missing comments, program lack

ns				structured code, consistent variable naming and formatting, well Commented code. Code readability, clarity, and adherence to coding standards. **(1 Mark)**	readability and coding standards **(0 Mark)**

7. The marking criteria valid for question 4b, 5 and 6 b 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	Could not explain the	Could explain basic terms but	Could explain reasoning behind	Could explain reasoning behind	Could explain reasoning behind the code, including use of loops, conditions,

	(Assignment submitted but absent in viva)	reasoning behind the code. But answered only one viva question	not about algorithm. But answered only two viva question	the code, including use of loops, conditions, algorithms. answered only three viva question	the code, including use of loops, conditions, algorithms. answered only four viva question	algorithms. Answered all five questions