Softwarica College of IT & E-Commerce
STW210CT: Programming, Algorithms and Data Structures

in collaboration with

Softwarica College of IT & E-commerce | Coventry University

Assignment Brief 2023

Module Name STW5008CEM: Programming for Developers	Ind/Group **Individual**	Cohort **Sep 2023**	Module Code: STW5008CEM
Coursework Title (e.g., CWK)			Hand out date: TBD
Lecturer: Hikmat Saud			Due date: TBD
Estimated Time (hrs): Word Limit*: n/a	Coursework type: Individual / Practical		% of Module Mark 33%
Submission arrangement online via Softwarica Moodle: 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:

1. Write software to solve a range of problems.
2. Implement and use simple searching and sorting algorithms.
3. Use libraries to extend the functionality of the base language.
4. Use basic design and testing strategies

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.

Question 1

a)

You are a planner working on organizing a series of events in a row of n venues. Each venue can be decorated with one of the k available themes. However, adjacent venues should not have the same theme. The cost of decorating each venue with a certain theme varies.

The costs of decorating each venue with a specific theme are represented by an n x k cost matrix. For example, costs [0][0] represents the cost of decorating venue 0 with theme 0, and costs[1][2] represents the cost of decorating venue 1 with theme 2. Your task is to find the minimum cost to decorate all the venues while adhering to the adjacency constraint.

For example, given the input costs = [[1, 5, 3], [2, 9, 4]], the minimum cost to decorate all the venues is 5. One possible arrangement is decorating venue 0 with theme 0 and venue 1 with theme 2, resulting in a minimum cost of 1 + 4 = 5. Alternatively, decorating venue 0 with theme 2 and venue 1 with theme 0 also yields a minimum cost of 3 + 2 = 5.

Write a function that takes the cost matrix as input and returns the minimum cost to decorate all the venues while satisfying the adjacency constraint.

Please note that the costs are positive integers.

Example: Input: [[1, 3, 2], [4, 6, 8], [3, 1, 5]] Output: 7
Explanation: Decorate venue 0 with theme 0, venue 1 with theme 1, and venue 2 with theme 0. Minimum cost: 1 + 6 + 1 = 7.

[5 Marks]

b)

You are the captain of a spaceship and you have been assigned a mission to explore a distant galaxy. Your spaceship is equipped with a set of engines, where each engine represented by a block. Each engine requires a specific amount of time to be built and can only be built by one engineer.

Your task is to determine the minimum time needed to build all the engines using the available engineers. The engineers can either work on building an engine or split into two engineers, with each engineer sharing the workload equally. Both decisions incur a time cost.

The time cost of splitting one engineer into two engineers is given as an integer split. Note that if two engineers split at the same time, they split in parallel so the cost would be split.

Your goal is to calculate the minimum time needed to build all the engines, considering the time cost of splitting engineers.

Input: engines= [1,2,3]

Split cost (k)=1

Output: 4

Example:

Imagine you need to build engine represented by an array [1,2,3] where ith element of an array i.e a[i] represents unit time to build ith engine and the split cost is 1. Initially, there is only one engineer available.

The optimal strategy is as follows:

1. The engineer splits into two engineers, increasing the total count to two. (Split Time: 1) and assign first engineer to build third engine i.e. which will take 3 unit of time.
2. Again, split second engineer into two (split time :1) and assign them to build first and second engine respectively.

Therefore, the minimum time needed to build all the engines using optimal decisions on splitting engineers and assigning them to engines. =1+ max (3, 1 + max (1, 2)) = 4.

Note: The splitting process occurs in parallel, and the goal is to minimize the total time required to build all the engines using the available engineers while considering the time cost of splitting.

[5 Marks]

Question 2

a)

You are the manager of a clothing manufacturing factory with a production line of super sewing machines. The production line consists of n super sewing machines placed in a line. Initially, each sewing machine has a certain number of dresses or is empty.

For each move, you can select any m ($1 <= m <= n$) consecutive sewing machines on the production line and pass one dress from each selected sewing machine to its adjacent sewing machine simultaneously.

Your goal is to equalize the number of dresses in all the sewing machines on the production line. You need to determine the minimum number of moves required to achieve this goal. If it is not possible to equalize the number of dresses, return -1.

Input: [1,0,5]

Output: 2

Example 1:

Imagine you have a production line with the following number of dresses in each sewing machine: [1,0,5]. The production line has 5 sewing machines.

Here's how the process works:

1. Initial state: [1,0,5]
2. Move 1: Pass one dress from the third sewing machine to the first sewing machine, resulting in [1,1,4]
3. Move 2: Pass one dress from the second sewing machine to the first sewing machine, and from third to first sewing Machine [2,1,3]
4. Move 3: Pass one dress from the third sewing machine to the second sewing machine, resulting in [2,2,2]

After these 3 moves, the number of dresses in each sewing machine is equalized to 2. Therefore, the minimum number of moves required to equalize the number of dresses is 3.

[5 Marks]

b)

You are given an integer n representing the total number of individuals. Each individual is identified by a unique ID from 0 to n-1. The individuals have a unique secret that they can share with others.

The secret-sharing process begins with person 0, who initially possesses the secret. Person 0 can share the secret with any number of individuals simultaneously during specific time intervals. Each time interval is represented by a tuple (start, end) where start and end are non-negative integers indicating the start and end times of the interval.

You need to determine the set of individuals who will eventually know the secret after all the possible secret-sharing intervals have occurred.

Example:

Input: n = 5, intervals = [(0, 2), (1, 3), (2, 4)], firstPerson = 0

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

Explanation:
In this scenario, we have 5 individuals labeled from 0 to 4.

The secret-sharing process starts with person 0, who has the secret at time 0. At time 0, person 0 can share the secret with any other person. Similarly, at time 1, person 0 can also share the secret. At time 2, person 0 shares the secret again, and so on.

Given the intervals [(0, 2), (1, 3), (2, 4)], we can observe that during these intervals, person 0 shares the secret with every other individual at least once.

Hence, after all the secret-sharing intervals, individuals 0, 1, 2, 3, and 4 will eventually know the secret.

[5 Marks]

Question 3

a) You are developing a student score tracking system that keeps track of scores from different assignments. The ScoreTracker class will be used to calculate the median score from the stream of assignment scores. The class should have the following methods:

- ScoreTracker() initializes a new ScoreTracker object.
- void addScore(double score) adds a new assignment score to the data stream.
- double getMedianScore() returns the median of all the assignment scores in the data stream. If the number of scores is even, the median should be the average of the two middle scores.

Input:

```
ScoreTracker scoreTracker = new ScoreTracker();
scoreTracker.addScore(85.5);    // Stream: [85.5]
scoreTracker.addScore(92.3);    // Stream: [85.5, 92.3]
scoreTracker.addScore(77.8);    // Stream: [85.5, 92.3, 77.8]
scoreTracker.addScore(90.1);    // Stream: [85.5, 92.3, 77.8, 90.1]
double median1 = scoreTracker.getMedianScore(); // Output: 87.8  (average of 90.1 and 85.5)

scoreTracker.addScore(81.2);    // Stream: [85.5, 92.3, 77.8, 90.1, 81.2]
scoreTracker.addScore(88.7);    // Stream: [85.5, 92.3, 77.8, 90.1, 81.2, 88.7]
double median2 = scoreTracker.getMedianScore(); // Output: 87.1 (average of 88.7 and 85.5)
```

[5 Marks]

b) Implement Kruskal algorithm and priority queue using minimum heap

[5 Marks]

Question 4

a)

You are given a 2D grid representing a maze in a virtual game world. The grid is of size m x n and consists of different types of cells:

'P' represents an empty path where you can move freely. 'W' represents a wall that you cannot pass through. 'S' represents the starting point. Lowercase letters represent hidden keys. Uppercase letters represent locked doors.

You start at the starting point 'S' and can move in any of the four cardinal directions (up, down, left, right) to adjacent cells. However, you cannot walk through walls ('W').

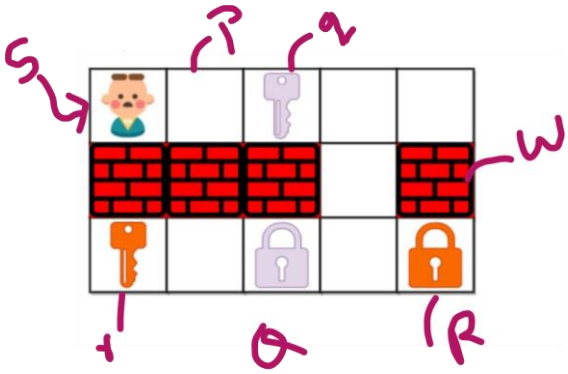
As you explore the maze, you may come across hidden keys represented by lowercase letters. To unlock a door represented by an uppercase letter, you need to collect the corresponding key first. Once you have a key, you can pass through the corresponding locked door.

For some 1 <= k <= 6, there is exactly one lowercase and one uppercase letter of the first k letters of the English alphabet in the maze. This means that there is exactly one key for each door, and one door for each key. The letters used to represent the keys and doors follow the English alphabet order.

Your task is to find the minimum number of moves required to collect all the keys. If it is impossible to collect all the keys and reach the exit, return -1.

Example:

Input: grid = [["S","P","q","P","P"], ["W","W","W","P","W"], ["r","P","Q","P","R"]]



Output: 8

The goal is to Collect all key

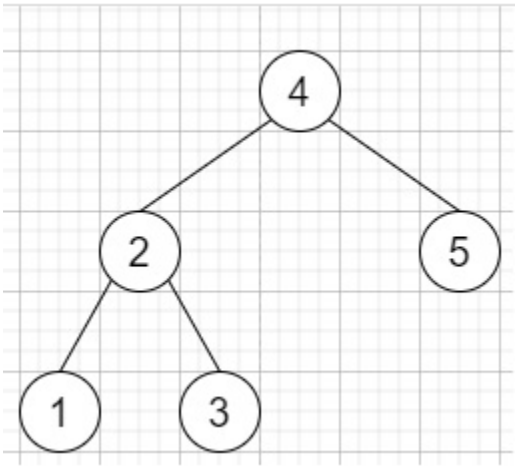
[5 Marks]

b)

You are provided with balanced binary tree with the target value k. return x number of values that are closest to the given target k. provide solution in O(n)

Note: You have only one set of unique values x in binary search tree that are closest to the target.

Input:



K=3.8

x=2

Output: 3,4

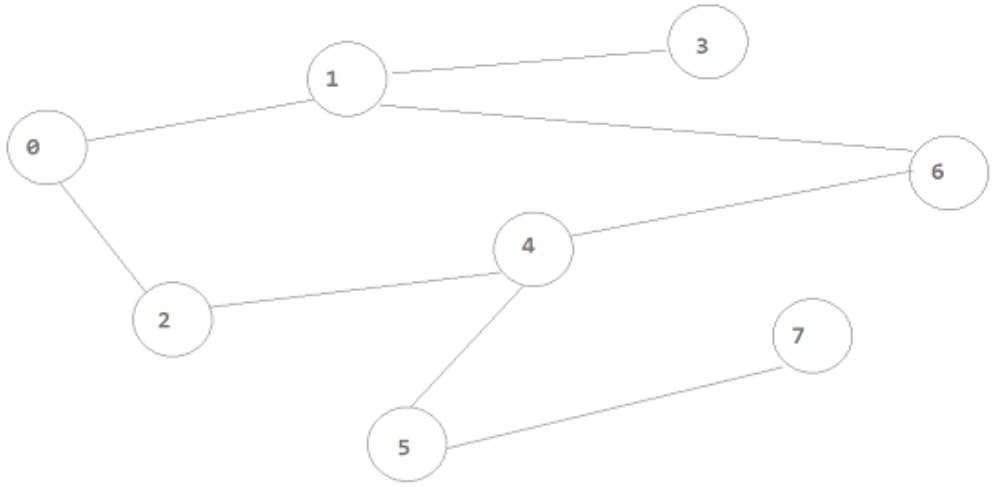
[5 Marks]

Task 5

a) Implement ant colony algorithm solving travelling a salesman problem

[5 Marks]

b) Assume you were hired to create an application for an ISP, and there are n network devices, such as routers, that are linked together to provide internet access to users. You are given a 2D array that represents network connections between these network devices. write an algorithm to return impacted network devices, If there is a power outage on a certain device, these impacted device list assist you notify linked consumers that there is a power outage and it will take some time to rectify an issue.



Input: edges= {{0,1},{0,2},{1,3},{1,6},{2,4},{4,6},{4,5},{5,7}}
Target Device (On which power Failure occurred): 4
Output (Impacted Device List) = {5,7}

[5 Marks]

Task 6

Implement a Multithreaded Asynchronous Image Downloader in Java Swing

Task Description:

You are tasked with designing and implementing a multithreaded asynchronous image downloader in a Java Swing application. The application should allow users to enter a URL and download images from that URL in the background, while keeping the UI responsive. The image downloader should utilize multithreading and provide a smooth user experience when downloading images.

Requirements:

Design and implement a GUI application that allows users to enter a URL and download images.

Implement a multithreaded asynchronous framework to handle the image downloading process in the background.

Provide a user interface that displays the progress of each image download, including the current download status and completion percentage.

Utilize a thread pool to manage the concurrent downloading of multiple images, ensuring efficient use of system resources.

Implement a mechanism to handle downloading errors or exceptions, displaying appropriate error messages to the user.

Use thread synchronization mechanisms, such as locks or semaphores, to ensure data integrity and avoid conflicts during image downloading.

Provide options for the user to pause, resume, or cancel image downloads.

Test the application with various URLs containing multiple images to verify its functionality and responsiveness.

Include proper error handling and reporting for cases such as invalid URLs or network failures

[20 Marks]

Task 7

Assignment: Graph-Based Recommendation System for a Social Media Application

Task Description:

You are tasked with designing and implementing a graph-based recommendation system for a social media application, inspired by platforms like TikTok. The application should utilize graph algorithms and user interactions to provide personalized content recommendations to users based on their interests and network connections.

Requirements:

Design and implement a graphical user interface (GUI) using a suitable GUI framework, such as Java Swing or JavaFX, to create an interactive social media application.

Implement a graph data structure that represents the social network, with nodes representing users and edges representing connections or interactions between users.

Provide functionality for users to create accounts, follow other users, and interact with content (e.g., liking, commenting, sharing).

Implement an algorithm to track user interactions and build a user profile based on their interests, preferences, and network connections.

Design and implement a recommendation algorithm that utilizes graph traversal and analysis techniques to suggest personalized content to users.

Consider factors such as user preferences, content popularity, and network influence in the recommendation algorithm to provide diverse and engaging recommendations.

Implement a user interface that displays recommended content to users based on their profile and interactions.

Allow users to provide feedback on recommended content, such as liking or disliking, to further refine the recommendation system.

Optimize the recommendation algorithm and data structures to handle large-scale social networks efficiently, considering factors such as memory usage and computational complexity.

Test the application with a simulated user base and interactions to evaluate the effectiveness of the recommendation system and the responsiveness of the GUI.

Consider implementing additional features to enhance the social media application, such as content filtering, user search, or real-time content updates.

Grading Criteria:

1. The application should meet all the requirements mentioned above.
2. The user interface should be intuitive and easy to use.
3. The application should be bug-free and stable.
4. The application should be well-documented and commented.
5. Bonus points will be given for additional features, such as algorithms to find the most influential users or to calculate the shortest path between two users.

Submission:

Submit the source code of the application along with a short report describing the features and functionalities of the application. The report should also include any known bugs or issues and suggestions for future improvements. The code should be well-organized and properly commented.

[30 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 Softwarica LMS.
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 valid for week 8-11 is presented below.

Criteria	0	1	2	3	4	5
Feature complete (10)	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 (10)	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 (10)	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.
Unit Testing (10)	Not submitted	Only a few features are tested without using framework and many of them fail	Many of the modules are tested and many of them fail	Many of the modules are tested using suitable unit testing framework.	Most of the modules are tested using suitable unit testing framework. Should have partial coverage.	All modules are unit tested using suitable unit testing framework. Should have full testing coverage.
Viva (10)	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