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

in collaboration with

Assignment Brief 2023

| | | | |
|---|---|---|---|
| Module Name<br>STW5008CEM: Programming for Developers | Ind/Group<br>**Individual** | Cohort<br>**Marh 2023** | Module Code:<br>STW5008CEM |
| Coursework Title (e.g., CWK) | | | Hand out date: TBD |
| Lecturer: Hikmat Saud | | | Due date: TBD |
| Estimated Time (hrs):<br><br>Word Limit*: n/a | Coursework type:<br>Individual / Practical | | % of Module Mark<br>25% |
| Submission arrangement online via Softwarica Moodle: Upload through Assignment links<br>File types and method of recording: URLs (source code repositories) Mark and<br>Feedback date: Within 3 weeks of assignment submission<br>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)

A trio of friends planned to purchase clothing from a particular store for an upcoming party, intending to wear matching outfits in varying colors - black, blue, and pink. The store had three different clothing sets on display, each in a different color. The shopkeeper assisted the three friends by selecting a clothing set in the appropriate color for each person based on their body shape and size. Given a 2D array, price[][3], where price[i][0], price[i][1], and price[i][2] represent the price of each clothing set for a different colored outfit for person i, your objective is to determine the minimum cost required to purchase clothing such that each person wears  have different color clothes if they stand in a row.

It should be noted that any two people can wear the same color cloth, but the third person must wear various color cloths, and all three can wear different colored garments.

***Input:*** *N = 3, price[][3] = [{14, 4, 11}, {11, 14, 3}, {14, 2, 10}]*
***Output:*** *9*
***Explanation:***
*Person 1 chooses blue clothing cost=4. Person 2 chooses pink clothing. Cost = 3. Person 3 chooses blue clothing. Cost = 2.*

*As a result, the total cost = 2 + 5 + 3 = 9.*
*Note: algorithm must take*
Time Complexity: O(N)
Auxiliary Space: O(1)

**[5 Marks]**

b)

A group of n Pathao riders stood in a queue, and each rider was assigned a unique integer rating based on their performance over the year. The Pathao company planned to distribute gold coins to each rider in ascending order, starting from count 1. The riders with higher ratings should receive more coins than their neighboring riders. The objective was to determine the minimum number of coins required by Pathao to distribute coins to the selected riders according to their ratings.

Input: ratings = [1,0,2]

Output: 5

Explanation: You can give the first, second, and third rider 2, 1, 2 gold coins, respectively.

**[5 Marks]**

**Question 2**

a)

Given an integer array nums and another integer k, the goal is to find the longest subsequence of nums that satisfies the following two conditions:

The subsequence is strictly decreasing.

The difference between adjacent elements in the subsequence is at most k.

The output should be the length of the longest subsequence that meets these requirements.

For example, consider the following input:

nums = [8,5,4, 2, 1, 4, 3, 4, 3, 1, 15] k = 3

output=[8,5,4,2,1] or [8,5,4,3,1]

Output: 5

Explanation:

The longest subsequence that meets the requirements is [8,5,4,2,1] or [8,5,4,3,1].

The subsequence has a length of 5, so we return 5.

Note that the subsequence [1,3,4,5,8,15] does not meet the requirements because 15 - 8 = 7 is larger than 3.

**[5 Marks]**

b)

Given an integer value k and an array of integers representing blacklisted ports, create an algorithm that outputs a random port (an integer value between 0 and k-1) that is also a whitelisted port (meaning it is not in the array of blacklisted ports). The goal is to minimize the number of built-in random calls in the algorithm.

The program should have two inputs: k, an integer value, and blacklisted_ports, an array of integers. The program should also have a get() function that returns a whitelisted random number between 0 and k-1. The algorithm should be optimized to reduce the number of built-in random calls required.

Example 1:

Input

["Program", "get", "get "get", "get", "get"]

[[7, [2, 3, 5]], [], [], [], [], [], [], []]

Output

[null, 0, 4, 6,1,4]

Explanation

program p = new program(7, [2, 3, 5]);

p.get(); // return 0, any number from [0,1,4,6] should be ok. Note that for every call of pick,

// 0, 1, 4, and 6 must be equally likely to be returned (i.e., with probability 1/4).

p.get(); // return 4

[5 Marks]

**Question 3**

a) Suppose you are provided an array of n targets that are placed in a row from 0 to n-1. Each target is assigned with certain integer such that a [0] represent the value associated with target zero. You are asked to shoot all the targets. If you shoot I th target then you will get a[i-1]*a[i]*a[i+1] points.

Note that if i-1 and i+1 position hits index out of bound, then you can assume that two target with value 1 are padded before start target and after end target.

Return maximum point one can gain by hitting each target.

**Input:** a = [3,1,5,8]

**Output:** 167

**Explanation:**

a = [3,1,5,8]

[3,1,5,8]  points 3*1*5 (“hitting target with value 1”)

[3,5,8] points 3*5*8 (“hitting target with value 5”)

[3,8]  points 1*3*8 (“hitting target with value 3”) note that there is padded target with value 1 at beginning and end of the provided target list

,[8]  points 1*8*1 same as above

points =      3*1*5+   3*5*8   +  1*3*8  + 1*8*1 = 167

**[5 Marks]**

b) Implement bellman ford algorithm and priority queue using maximum heap

**[5 Marks]**

**Question 4**

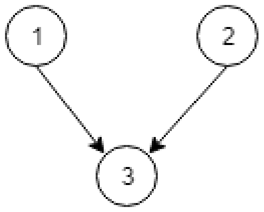
a)

There are n tasks you need to complete for a game, labelled from 1 to n.

We are given r[i]=[x,y] representing a prerequisite relationship between task x and task y: task x has to be completed before task y.

In one step you can complete any number of task as long as you have completed all the prerequisites for the tasks you are provided while playing game.

Return the minimum number of steps needed to complete all tasks.  If there is no way to complete all the tasks, return -1.



Input: N = 3, r= [[1,3],[2,3]]
Output: 2
Explanation:
In the first step, you can complete task 1 and 2. In the second semester, step task 3 can be completed.

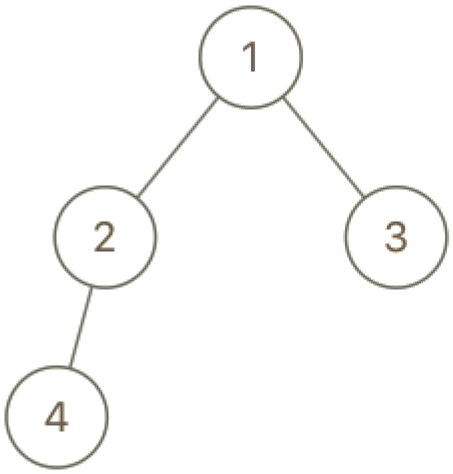
**[5 Marks]**

b)

Given the root of a binary tree with unique values and the values of two different nodes of the tree x and y, return true *if the nodes corresponding to the values* x *and* y *in the tree are* **brothers***, or* false *otherwise.*

Two nodes of a binary tree are brothers if they have the same depth with different parents.

Note that in a binary tree, the root node is at the depth 0, and children of each depth k node are at the depth k + 1.



**Input:** root = [1,2,3,4], x = 4, y = 3

**Output:** false

**[5 Marks]**

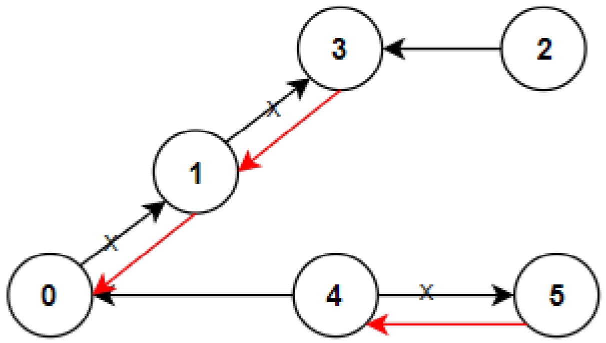
Task 5

a) Implement hill climbing algorithm

**[5 Marks]**

b)

A network consisting of n servers is connected in a tree structure, where the servers are numbered from 0 to n - 1 and there are n - 1 connections between them that only allow for one-way communication. A 2D array a is used to represent these connections, where a[i] = [ai, bi] represents a directed path from server ai to server bi. However, due to specific requirements, all traffic from each server must route to server 0. The task is to reorient some connections to ensure that each server has a path to server 0. The goal is to minimize the number of edges that need to be changed. It is guaranteed that every server must have a path to server 0 after the connections are reordered.



**Input:** n = 6, connections = [[0,1],[1,3],[2,3],[4,0],[4,5]]

**Output:** 3

**Explanation:** Change the direction of edges show in red such that each node can reach the node 0.

[5 Marks]

Task 6

Write a Java program that uses multithreading to implement a parallel merge sort algorithm. Your program should meet the following requirements:

1. Input: Your program should accept an array of integers as input.
2. Output: Your program should output the sorted array.
3. Algorithm: Your program should use a parallel merge sort algorithm to sort the input array. The algorithm should divide the input array into subarrays, sort the subarrays in parallel using multiple threads, and then merge the sorted subarrays to produce the final sorted array.
4. Performance: Your program should be optimized for performance, such that it sorts the input array as quickly as possible. You should experiment with different thread counts and input array sizes to find the optimal settings.
5. Error handling: Your program should handle errors and exceptions gracefully, such as by providing informative error messages and exiting gracefully.
6. Testing: Your program should be thoroughly tested to ensure that it correctly sorts the input array and produces the expected output.

To complete this assignment, you will need to implement the parallel merge sort algorithm in Java using multithreading. You should also experiment with different thread counts and input array sizes to find the optimal settings for performance. You can use Java's built-in threading and synchronization features, such as the Thread class and synchronized keyword, to implement the parallel merge sort algorithm.

**[20 Marks]**

Task 7

Assignment Title: Social Network Graph

Task: Create a GUI application that allows users to visualize a social network graph.

Scenario: You have been hired by a social media company to create a tool that visualizes the connections between users. The company wants to see how users are connected and which users have the most influence over others.

Requirements:

1. The application should have a window with a canvas where the graph will be drawn.
2. The nodes of the graph should represent users, and the edges should represent connections between users.
3. The application should read the user data from a file and create the graph accordingly.
4. Each node should display the user's name, profile picture, and the number of followers they have.
5. The edges should display the strength of the connection between the users, such as the number of likes, comments, or shares between them.
6. The user should be able to select and move nodes around the canvas.
7. The user should be able to delete nodes and edges by selecting them and pressing the delete key.
8. The application should have a toolbar with buttons for selecting mode, adding nodes, and adding edges.
9. The application should allow the user to search for a user and highlight their node and connections.

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 |