BRAC UNIVERSITY
Department of Computer Science and Engineering

CSE220: Data Structures

A

Examination: Final
Duration: 1 Hour 40 Minutes

Semester: Spring 2025
Full Marks: 35
No. of Pages: 4

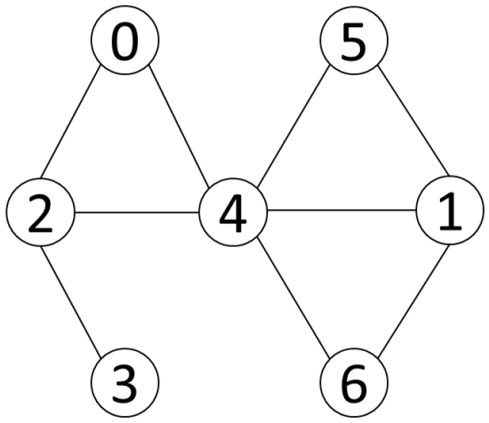
Name: (Please write in CAPITAL LETTERS)	ID:	Section:

✔ **Answer all questions. No washroom breaks. Understanding questions is part of the Exam.**
✔ At the end of the exam, put the question **paper** inside the answer script and **return both**.

Question 1: CO1 [5 Points]

Write the correct answer (only option if MCQ) in your answer script. If you think there are multiple answers, then write all the answers.

i. Which one of the following sequences is a possible order of visiting the nodes in the graph below if we use BFS traversal?



A. 0, 5, 2, 4, 1, 3, 6
B. 4, 6, 1, 5, 3, 2, 0
C. 0, 4, 5, 1, 6, 2, 3
D. 1, 4, 5, 6, 0, 2, 3

ii. A heap is always a balanced binary tree.
A. True
B. False

iii. Find the correct statements
A. In a Perfect Binary Tree of height x, the total number of leaf nodes is 2^x.
B. In a Complete Binary Tree, if a node doesn't have a right child, it must not have a left child as well.
C. In a Binary Tree, consider two sibling nodes: A, B. The level of A = 3, and the level of B = 2.
D. All Binary Search Trees are Binary Trees, but not all Binary Trees are Binary Search Trees.

iv. The maximum number of swaps needed to heapify a single value in a heap of height *h* _____.
v. A Binary Search Tree is considered balanced if the height difference between the left and right subtrees of every node is at most **_____**.

Question 2: CO3 [10 Points]

You are given two heads of a singly linked list that may or may not intersect at some node. If they do intersect, they share a common tail segment (i.e., from the intersecting node onward, both lists are identical in structure and content). Your task is to write a method called **mergeLL(h1, h2)** that takes the two heads as parameters and **returns the merged head**.

This method should **merge the second linked list into the first**, only if the lists intersect. The resulting merged list should be a single list that starts from h1 and includes all nodes, avoiding any duplication of the overlapping segment.

If there's an intersection, the merge should connect the non-overlapping portion of h1 to the head of h2, up to the point where both lists start sharing nodes. If there's no intersection, then the program will return None/null. The Node class has an elem variable (Integer type) and a next variable (Node type).

Restrictions:

- **Do not create any new nodes.**
- **Do not modify the values in the existing nodes.**
- **Do not create any instances of any other data structures (e.g, array, stack, etc.)**

Sample Given Linked List	Returned Linked List
An intersected linked list example	
(h1) *56 → 78 → 91* ↘ ***62 → 17 → 89 → 24*** ↗ (h2) ***43 → 33***	*56→78→91→**43→33→62→17→89→24***
Non-intersecting linked list example	
(h1) 56 → 78 → 91→17 → 89 → 24 (h2) 43 → 33 → 36 → 29	*None/null*

Python Notation	Java Notation

`def mergeLL(h1, h2):`
`  # To Do`
	`static Node mergeLL(Node h1, Node h2){`
	`  // To Do`
	`}`

Question 3: CO4 [8 Points]

Complete the function below, which will receive the root of a binary tree as a parameter and return the second-highest value from the tree. Assume there will always be a second-highest value in the tree. **The node class has an elem (Integer type), left (Node type), and right (Node type) variable.**

Restrictions:

- You cannot use any global variables and cannot create any other data structures (e.g, Array length more than 2, linked list, etc).
- You cannot change the number of parameters of the second_max function.
- You can design and use helper functions. You can use an array of length 2 if you want to.

Input tree	Returned Value	Explanation
30 / \ 50 10 / 40	40	In the given binary tree, the maximum value is 50 and the second-highest value is 40.

Python Notation	Java Notation

`def second_max(root):`
`  #To do`
	`static int second_max(Node root){`
	`  //To do`
	`}`

Question – 4: CO3 [4 Points]

Complete the function below that will take an array of tasks and print out the completion step of tasks based on their priorities. The value of the index of an array will represent the priority of that task. The lower the value, the higher the priority. One task will be done in one step. **Assume that all the values of the array will be unique, and you have a magical function `getIndex(arr, val)` which will return the index of that value in the array in constant time.**

The time complexity of your solution must be less than $O(n^2)$. You may use the provided MaxHeap or MinHeap classes with the **`insert(item)`** and **`extract()`** (extract will return the extracted value) methods. **Check the sample input-output to find out the printing format.**

You can create an Empty Heap using the following code	
Python Notation	Java Notation
`max_heap = MaxHeap()` `min_heap = MinHeap()`	`MaxHeap maxHeap = new MaxHeap()` `MinHeap minHeap = new MinHeap()`

Python Notation	Java Notation

`def priority_task(tasks):`
`  #To do`
	`static void priority_task(int[ ] tasks){`
	`  //To do`
	`}`

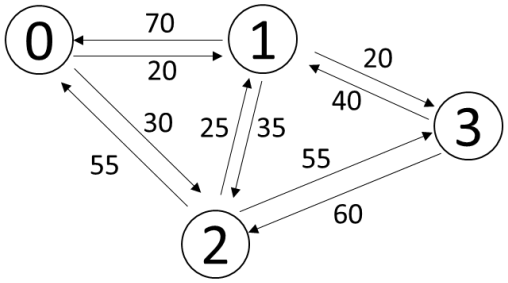
Input	Output	Explanation

`tasks = [60, 85, 70, 45]`	`Step 1 - Task 4`
	`Step 2 - Task 1`
	`Step 3 - Task 3`
	`Step 4 - Task 2`
		Here, index 3 has the minimum value 45, so it will be done at step 1. `getIndex(tasks, 45)` will return 3, hence the task number is (3 + 1) = 4. The rest of the output is generated following the same pattern.

Question – 5: CO2 [8 Points]

The National Transport Authority monitors traffic volumes between cities every day. They are using a network modeled as a directed, weighted graph, where:

- Vertices represent cities
- Directed edges represent one-way roads
- Edge weights represent the average number of vehicles per day traveling from the source city to the destination city



The authority now wants to identify the city that has the least total incoming traffic, which refers to the vertex with the minimum sum of incoming edge weights. For the given example in the figure, the output should be city 1.

Now write a method/function that takes a directed, weighted graph (adjacency list) as an input parameter and returns the city with the minimum total incoming traffic volume.
You can assume that the Edge class has source (Type: String), destination (Type: String), weight (Type: Integer), and a next (Type: Edge) variable. Note that the adjacency list represents the out-degree of the vertices.

Python Notation	Java Notation

`def find_minimum(graph):`
`  #write your code here`
	`static String find_minimum(Edge [] graph){`
	`  //write your code here`
	`}`

Sample Input	Output	Explanation
0 | 1 | 2 | 3 <0,1,20> <1,0,70> <2,0,55> <3,1,40> <0,2,30> <1,2,35> <2,1,25> <3,2,60> <1,3,20> <2,3,55>	1	The sum of the incoming edge weights for **vertex 1** is: 20+25+40 = 85 which is the minimum among all. Hence, 1 is returned.