A

BRAC UNIVERSITY
Department of Computer Science and Engineering

Examination: Mid-Term
Duration: 75 Minutes
Number of Questions: 3

CSE220: Data Structures

Semester: Fall 2024
Full Marks: 30
No. of Pages: 3

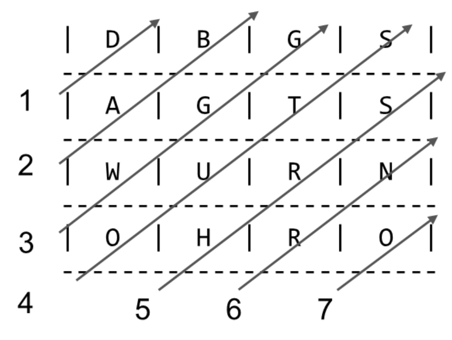
Name: (Please write in CAPITAL LETTERS)	ID:	Section:

- **Answer all 3 questions. No washroom breaks.**
- At the end of the exam, put the question **paper** inside the answer script and **return both**.
- For Java, just complete the function, no need to write the class and main method.

Question 1: CO1 [2 + 8 Points]

I. Suppose you are given a multi-dimensional array with dimensions 3x4x2. What is the linear index of the multidimensional index [2][1][0]?

II. You have hidden a password into a square matrix for your friend. Only you know that the password can be achieved through traversing the array in a zigzag fashion. Now help your friend develop a function/method that will take the 2D array as input and print the password in the output console.
Complete the function zig_zag which will take a NxN 2D array and print all the values in a zigzag pattern. Each value will be separated by a space. Assume the function will always receive a NxN 2D array. Your code should work for any NxN array.

Input Array	Output	Zigzag traversal explanation
| D | B | G | S | ------------------------- | A | G | T | S | ------------------------- | W | U | R | N | ------------------------- | O | H | R | O | -------------------------	D A B W G G O U T S H R S R N O	 D B G S A G T S W U R N O H R O 1 2 3 4 5 6 7

Hint: You need to follow the arrowed line 1 to line 7 to find out the zigzag pattern. Use one loop to print the upper triangle (line 1 to 4) and another loop to print the lower triangle (line 5 to 7).

Python Notation	Java Notation

`def zig_zag(arr):`
`     # To Do`
	`public void zig_zag(int[][] arr) {`
	`       // To Do`
	`}`

Question 2: CO5 [10 Points]

You are given a non-dummy-headed, singly linear linked list containing positive integers. Your task is to complete the given method **reverseAndSwap()** that takes the **head** of the linked list and an **integer i** as input and returns the head of the modified list.

Your task is to
- Reverse the list from index 0 to i
- Swap the two parts of the list, i.e., the unchanged part from index i + 1 to total_nodes – 1 will come before the reversed part from index 0 to i in the new list. (where total_nodes = number of nodes in the linked list)

Check the given input and output for more clarification.

Note: You can assume that the index i will always be in the range 0 to total_nodes – 1. Assume Node class has elem and next variable. No need to write the Node class.

[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LIST, i.e. YOU CANNOT CREATE A NEW LINKED LIST]

Sample Input	Sample Output	Explanation
5→7→6→3→8→2→1 i = 3	8→2→1→3→6→7→5	The list is reversed from index 0 to 3 producing 3→6→7→5. The unchanged part (8→2→1) now sits before the reversed part, producing the output list.
5→7→6→3→8→2→1 i = 0	7→6→3→8→2→1→5	The list is reversed in index 0, producing 5. The unchanged part (7→6→3→8→2→1) now sits before the reversed part, producing the output list.
5→7→6→3→8→2→1 i = 6	1→2→8→3→6→7→5	The list is reversed from index 0 to 6, there is no part from the original list left.
5→7→6→3→8→2→1 i = 5	1→2→8→3→6→7→5	The list is reversed from index 0 to 5 producing 2→8→3→6→7→5. The only node remaining (Node 1) comes before the reversed part producing the output list.

Python Notation	Java Notation

`def reverseAndSwap(head, i):`
`    # To Do`
	`public Node reverseAndSwap(Node head, int i) {`
	`    // To Do`
	`}`

Question 3: CO3 [10 Points]

You are given the start and end time of some tasks. The tasks are sorted according to their starting time. You can merge two tasks **A** and **B if they are overlapping.** That is **task B** starts before the finishing time of **task A**. In that case, it becomes a single task having the ending time of **task A** or **task B** (depending upon which task finishes later). This task can be merged further with other tasks if the above condition satisfies (overlapping). You want to merge as many tasks as possible and minimize the total number of non-overlapping tasks.

You are given a 2D array named *tasks* storing the start and end time of the task. The array is sorted according to the starting time of the tasks. Each row of the array stores two integer values. The first value indicates the start time and the second value indicates the end time of a task. More specifically, the start and end time of the ***i**th* task is `tasks[i][0]` and `tasks[i][1]` respectively.

Write a function named ***`print_total_task(tasks)`*** that takes the tasks array as input and prints the start time and end time for each non-overlapping task according to the given format. Note that the tasks are printed as decreasing order of start time.

Consider that the Stack class is already available with the following methods/functions implemented: push(element), pop(), peek(), and isEmpty(). Stack Underflow and Stack Overflow exceptions return None (Python) / null(Java). There is no need to write the Stack class.

You are allowed to create as many *stacks* as you want. You cannot create any new array. To create a Stack object use the following:
stack = Stack() #(Python)   Stack stack = new Stack() //(Java)

Example 1:	**Explanation:**
Input: *tasks* = [[1, 5], [2, 3], [4, 6], [7, 10], [9, 11], [12, 15]] **Output:** 12, 15 7, 11 1, 6	Tasks (1,5) and (2,3) are overlapping, so we can merge them creating the new task (1,5) Then task (1,5) and (4,6) are also overlapping, merging them creates the new task (1,6) We can not merge tasks (1,6) and (7,10) as they are non-overlapping However tasks (7,10) and (9,11) can be merged creating new task (7, 11) Finally tasks (7,11) and (12,15) are non-overlapping So we have 3 non-overlapping tasks (1,6), (7,11) and (12,15) Tasks are printed in descending order of start time.

Python Notation	Java Notation

`def print_total_task(tasks):`
`        # To Do`
	`public void print_total_task(int [ ][ ] tasks){`
	`        // To Do`
	`}`