

Examination: Mid-Term
Duration: 80 Minutes
Number of Questions: 3

CSE220: Data Structures

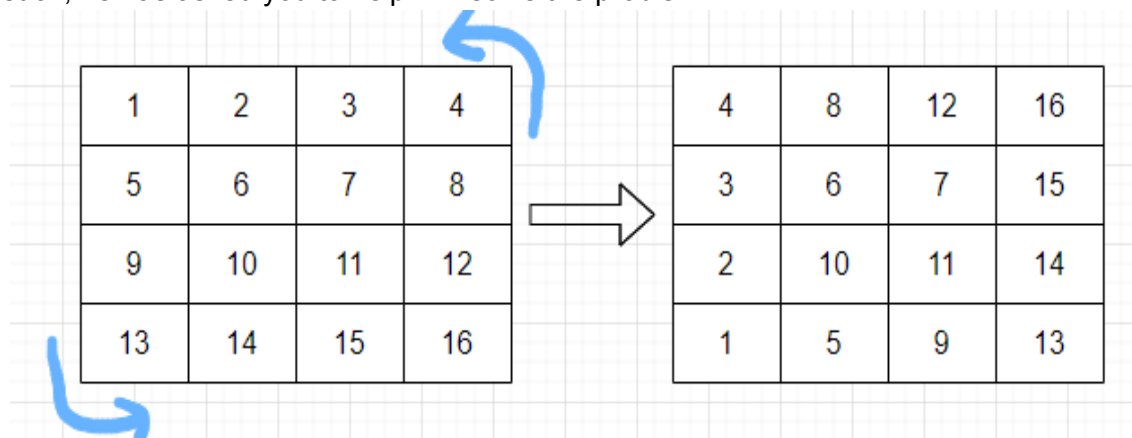
Semester: Summer
 2024
Full Marks: 35
No. of Pages: 4

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.**

Question 1: CO1 [2 + 8 Points]

- Suppose you are given a multi-dimensional array with dimensions 4x3x2x6. What is the corresponding linear index of the multidimensional index [2][1][0][4]?
- Bob has already learnt about 2D matrices and its operations in the data structures course. However, he is currently facing problems in rotating a 2D square matrix. As such, he has asked you to help him solve the problem.



You should rotate the original input matrix's outer rows and columns in left, down, right, top manner without affecting the internal cells; **instead of creating a new matrix.**

Here, as you can see, if you rotate the given square matrix **anticlockwise**, 1st row will be 1st column, 1st column will be last row, last row will be last column and last column will be 1st row. The elements of the internal cells will be unchanged.

So given a square matrix of size $n \times n$, your task is to complete the given method **rotate(matrix)** that takes a 2D square matrix as an input and returns the rotated matrix using **in-place** algorithm.

Hints of in-place: Rotating by one element at a time instead of an entire row/column should eliminate the need for any supporting data structures.

Sample Input	Sample Output
<pre> 1 2 3 ----- 5 6 7 ----- 9 10 11 ----- </pre>	<pre> 3 7 11 ----- 2 6 10 ----- 1 5 9 ----- </pre>

Sample Input	Sample Output
<pre> 1 2 3 4 ----- 5 6 7 8 ----- 9 10 11 12 ----- 13 14 15 16 ----- </pre>	<pre> 4 8 12 16 ----- 3 6 7 15 ----- 2 10 11 14 ----- 1 5 9 13 ----- </pre>

Python Notation	Java Notation
<pre> def rotate(matrix): # To Do </pre>	<pre> public int[][] rotate(int[][] matrix) { // To Do } </pre>

Question 2: CO5 [10 Points]

Suppose, you are working on a web application which has a customer support system. You are assigned to handle the customer support tickets. The support tickets are stored in a non-dummy headed singly linear linked list on a first-come first-served basis by IDs. However, due to a glitch, some tickets have been duplicated. Now, your task is to complete the given method **remove_Duplicates(head)** which takes the head of the linked list as input to remove the duplicate IDs ensuring that each ticket appears only once.

[DO NOT USE OTHER DATA STRUCTURE OTHER THAN GIVEN LINKED LIST]

Sample Input	Sample Output	Explanation
Input Tickets: 101 -> 103 -> 101 -> 102 -> 103 -> 104 -> 105 -> 105	Fixed Tickets: 101 -> 103 -> 102 -> 104 -> 105	All the duplicates of 101, 103 and 105 have been removed.
Input Tickets: 102 -> 101 -> 101 -> 102 -> 102 -> 102 -> 103 -> 104 -> 104	Fixed Tickets: 102 -> 101 -> 103 -> 104	All the duplicates of 102, 101 and 104 have been removed.

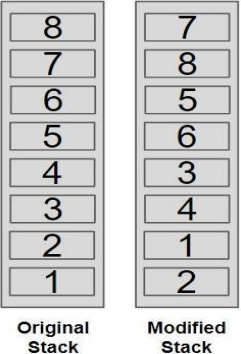
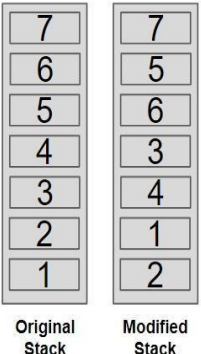
Python Notation	Java Notation
<pre> def remove_Duplicates(head): # To Do </pre>	<pre> public Node remove_Duplicates(Node head) { // To Do } </pre>

Question 3: CO3 [7+8 Points]

- I. You are given a stack of some integer elements, st. Your task is to swap the adjacent elements of the stack and return the modified stack. If the number of elements in the stack is odd, keep the topmost element as it is and swap the other adjacent elements. See the input output for better understanding. Now write a function **Do_Adjacent_Swap(st)** that completes the given task and returns the modified stack.

The st is an object of the “Stack” class which is implemented by a singly linear linked list and consider that the following methods of “Stack” class are already implemented for you: **push(element)**, **pop()**, **peek()**, **isEmpty()**.

You cannot use any other data structures other than stack. The “Top” pointer of Stack class is NOT accessible to you. You can use just the above four methods of Stack class. You can create as many stacks as you need for this task.

Sample Input	Sample Output	Explanation
Original Stack: 8(Top) → 7 → 6 → 5 → 4 → 3 → 2 → 1	Modified Stack: 7(Top) → 8 → 5 → 6 → 3 → 4 → 1 → 2	
Original Stack: 7(Top) → 6 → 5 → 4 → 3 → 2 → 1	Modified Stack: 7(Top) → 5 → 6 → 3 → 4 → 1 → 2	

Python Notation	Java Notation
<pre>def Do_Adjacent_Swap(st): # To Do</pre>	<pre>public Stack Do_Adjacent_Swap(Stack st): { // To Do }</pre>

- II. You are given a **circular array-based queue** where the length of the array is 4 and the starting index of front and back is 3.

You need to perform the queue operations based on the function given below. This function takes the head of a non-dummy-headed singly linear linked list and a queue object as an input.

Linked_List = -1 → 5 → -12 → -20 → 9 → -43 → -67 → -67 → -85 → -91 → -90

```
def test(head,queue):
    temp = head
    while temp.next != None:
        if temp.next.elem < temp.elem:
            queue.enqueue(temp.next.elem)
        elif temp.next.elem > temp.elem:
            queue.dequeue()
        else:
            queue.peek()
        temp = temp.next
```

Show the queue, front index and back index in the following table for each of the nodes.

The Print Message includes Queue Overflow, Queue Underflow, Peeked items and Dequeued items.

No need to write anything in the Print Message when the enqueue operation is performed.

Fillup the following table in your answer script. No need to write in the question paper.

Temp Node	0	1	2	3	Front	Back	Print Message
Initial					3	3	
Node -1							
Node 5							
Node -12							
Node -20							
Node 9							
Node -43							
Node -67							
Node -67							
Node -85							
Node -91							
Node -90	-	-	-	-	-	-	-