BRAC UNIVERSITY
Department of Computer Science and Engineering

A

Examination: Mid-Term
Duration: 75 Minutes
Number of Questions: 4

CSE220: Data Structures

Semester: Spring 2025
Full Marks: 30
No. of Pages: 3

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

- **Answer all 4 questions. No washroom breaks.**
- At the end of the exam, put the question **paper** inside the answer script and **return both**.

Question 1: CO1 [10 + 1 Points]

You are given a **9x5** dimensional 2D array which is used as a combination lock of a safe. You can independently rotate each column either upward or downward. You need to write a function that will take the 2D array and combination of the safe as an 1D array of length 5 and unlock the safe. To unlock the safe you need to make the **middle row (4th row** as rows start from 0) **same as the input 1D array.**

Note that the values of each column are unique and in the range of 1 to 9. To equate the middle row with the given combination (1D array), you need to perform **column rotation operation** (Just like an original safe). You may write and use helper functions.

The space complexity of your solution should be O(1) that means you cannot create any new array. **Finally, return the resulting array. In addition, write the Time complexity of your solution in terms of Big O notation.**

Input Array	Returned Array	Explanation
2 8 9 6 7 4 2 5 8 5 6 7 1 4 3 9 6 7 2 9 **7 9 3 5 6** 8 1 6 9 2 5 4 2 7 1 3 3 8 1 4 1 5 4 3 8 combination: 1 2 3 4 5	7 4 9 1 1 8 3 5 3 4 5 5 1 6 8 3 8 7 8 7 **1 2 3 4 5** 2 7 6 2 3 4 6 2 5 9 6 9 8 9 6 9 1 4 7 2	The returned array is rotated column wise in such a way so that the 4th row and the combination array have the same sequence of values. Column wise rotation may be done upward or downward. No need to implement both.

Python Notation	Java Notation

`def unlockSafe(mat, com):`
`  # To Do`
	`public int[][] unlockSafe(int[][] mat, int[] com){`
	`  // To Do`
	`}`

Question 2: CO5 [7 Points]

You are going to design a Hashtable with forward chaining. So each index of the hash table contains a non dummy headed singly linear linked list to resolve collision. **Your task is to implement the following function with the following requirements.**

- **insert_HashTable:** This function takes a key, a value, and the hash table as parameters and inserts the pair in its appropriate position. However, the insertion is not straightforward. It follows the following conditions.
  - If the key already exists, only update the value.
  - Otherwise
    - If the value is even: insert it at the beginning of the linked list.
    - If the value is odd: insert it at the end of the linked list.

The Node class for the hash table is already implemented with key (String), val (int), and next (Node) variable / attribute.

The hash function is already implemented as named **hash_Function**. You need to use it during insertion.. You need to use it during insertion.

# Python notation	// Java notation

def hash_Function(key):
  # use this function
  # already written
 	
def insert_HashTable(key, value, ht):
  # Write your code here
	int hash_Function(String key){
	  // use this function
	  // already written
	}
	
	void insert_HashTable(String key, int value, Node[ ] ht){
	  // Write your code here
	}

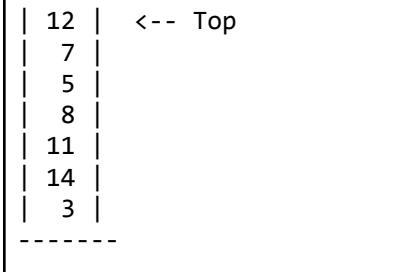
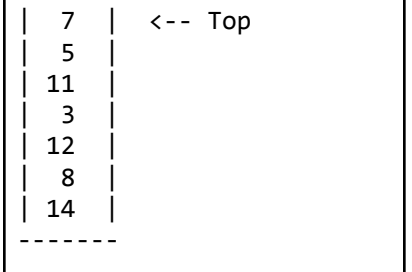
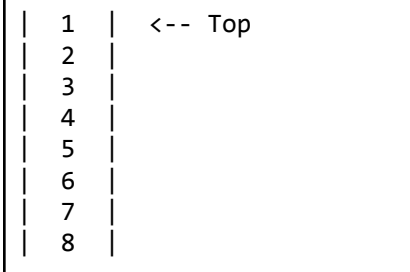
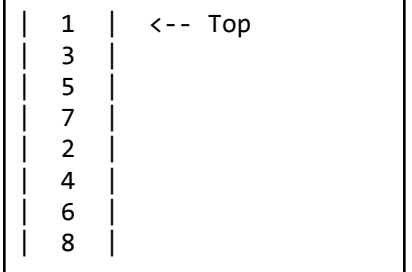
Question 3: CO3 [7 Points]

You are given a stack containing integers in arbitrary order. Your task is to write a function that segregates odd and even numbers while maintaining their relative order and finally returns the stack.

- Odd numbers should appear at the top (in the same order as in the original stack).
- Even numbers should appear at the bottom (in the same order as in the original stack).
- You can only use stack operations: push(value), pop(), peek(), and isEmpty().
- You may use multiple auxiliary stacks but cannot use other data structures like arrays or lists.
- For Overflow and Underflow exceptions, assume the methods return None (Python) / null (Java)

Stack Creation Example

Python	Java
`stack = Stack()`	`Stack stack = new Stack()`

Input Stack	Modified Stack	Explanation
		The odd numbers (7,5,11, 3) are grouped together and appear at the top of the output stack, in the same order as the original stack. The even numbers appear next, maintaining their original order.
		Same reasoning as before

Python Template	Java Template

`def segregate(original):`
`  // Write your code`
	`public Stack segregate(Stack original) {`
	`  // Write your code`
	`}`

Question 4: CO2 [2 + 1 + 1 + 1 Points]
(Write the correct answers in the answer script)

I. Suppose you are given a multi-dimensional array with dimension **5x5x3**. What is the multidimensional index for the linear index **25**?

II. What is the use of 'rear' or 'tail' in a Queue?
   A. Points to the first inserted element
   B. Helps in Dequeue() to remove elements from the front
   C. Helps in Enqueue() to add elements at the back
   D. Points to the element to be enqueued

III. To dequeue an element what should not be done?
   A. Check if the queue is already empty or not
   B. Remove the element that first came to the queue
   C. Remove the first node of the queue with a time complexity of O(1)
   D. Return the last element of the queue

IV. What does the peek() function do in a Queue?
   A. Return the value at the front of the queue
   B. Return the value from the first node and remove the node
   C. Return the value from the last node and remove the node
   D. Return the value at the end of the queue