BRAC UNIVERSITY
Department of Computer Science and Engineering

A

Examination: Final
Duration: 1 Hour 40 Minutes
Number of Questions: 5

CSE220: Data Structures

Semester: Fall 2023
Full Marks: 45
No. of Pages: 4

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

✔ **Answer all 5 questions. No washroom breaks.**
✔ 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 in your answer script:

I. arr = [2, 1, 5, 3, 5, 5, 4, 2, 1, 2, 5, 3].
An auxiliary array aux_arr is built using key-indexing.

Which of the following is TRUE?

a. aux_arr[2] = 1, b. aux_arr[1] = 0,
c. aux_arr[2] = 3, d. aux_arr[5] = 5

II. Look at the function:

```
def fun(n):
  if n < 2:
    return 3
  return fun(n-2) + 2 * fun(n-4)
```

What is the return value of fun(5)?

a. None, b. 15, c. 52, d. Stack Overflow

III. Look at the following array:
arr = [3, 1, 4, 2, 5]

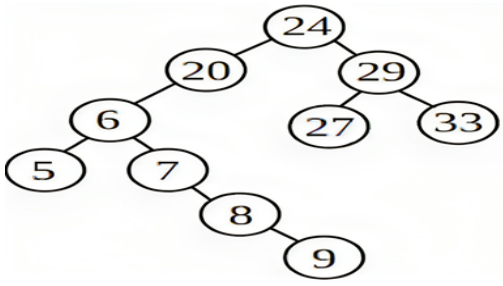
What will be the value of arr[arr[arr[3]]]?

a. None, b. 3, c. 5, d. Syntax Error

IV. Insertion in the middle of Dummy-Headed Doubly Circular Linked List requires modification of how many pointers?

a. 1, b. 2, c. 3, d. 4.

V. . Notice the following BST



On this BST, these following operations are done step by step:
i) 23 is inserted
ii) Node 29 is deleted using successor
iii) The ROOT is deleted using predecessor

Which of the following is NOT TRUE about the resulting tree?
a. Number of leaf nodes DECREASES in the resulting tree
b. Node 23 is the ROOT node in the resulting tree
c. There are 5 nodes with only one child in the resulting tree
d. Node 33 is NOT a LEAF node in the resulting tree

Question 2: CO4 [10 Points]

You are given a Linked List, LL1, and an array, dist. Write a method **sum_dist(list, arr)** that takes a Linked List and an array as parameters. This method sums the node elements in the linked list that are away from the head by the elements in the array and returns the sum. Assume the Node class has only elem and next variable. **No need to write Node class and driver code.**

Sample Input	Sample Output	Explanation
LL1 = 10--> 16 --> -5 --> 9 --> 3 --> 4 dist = [2, 0, 5, 2, 8] Function Call: print(sum_dist(LL1, dist))	4	Node Element away from the head at distance 2 = -5 Node Element away from the head at distance 0 = 10 Node Element away from the head at distance 5 = 4 Node Element away from the head at distance 8 = Doesn't Exist, Considered as 0 The sum is: -5+10+4+-5+0 = 4

Question 3: CO4 [5 Points]

Constructor of Queue class is given to you for understanding the initial front and back positions. Suppose, enqueue(elem), and dequeue() methods are implemented inside the Queue class.

```
class Queue:
  def __init__(self,capacity):
    self.circArr=[None]*capacity
    self.front=capacity-1
    self.back=capacity-1
```

Write down all the outputs of the following code snippet **(That means you should print the status of the circular queue, front index, back index after each iteration).**

```
# Driver code
queue=Queue(4)
arr=[10,20,31,40,53]
test(arr,queue)
```

```
def test(arr,queue):
  for i in range(len(arr)):
     if arr[i]%2==0:
          queue.enqueue(arr[i])
     else:
          queue.dequeue()
     print(queue.circArr) # Print the array
     print(queue.front)
     print(queue.back)
```

Question – 4: CO3 [2+8 Points]

Suppose you are given a hash function called check(). In this hashing, forward chaining is used for resolving conflict, and a 3-length array of singly linked lists is used as the hash table. In the singly linked list, each node has a next pointer, a string key and a string value, for example: ("7B12C" (key), "CSE220" (value)). The hash-table stores this key-value pair. The check function is given below:

For your reference ASCII value of A is 65.

```
Hash Function
def check(key):
   v = 0
   for i in key:
      if i.isnumeric():
         v+=int(i)
      else:
         v+=ord(i)
   return v%3
```

I. What is the hashed-index of ("4G14", "MNS") key-value pair in the above hashTable?

II. Implement a function **remove(hashTable, key)** that takes a key and a hash-table as parameters. The function removes the key-value pair from the aforementioned hashtable if such a key-value pair (whose key matches the key passed as argument) exists in the hashtable and return the table. If the key does not exist, the function returns the same hash table.

Consider, Node class, hash_function and hashTable are given to you. **You just need to complete the remove(hashTable, key) function.**

```
class Node:
def __init__(self, key, value, next=None):
        self.key, self.value, self.next = key, value, next
```

Python Notation:

```
def remove(hashTable, key):
        # To Do
```

Sample Input and Output

Input Table	Returned Table
0: (“13D”, “ZMD”) → (“10A”, “ABW”) 1: (“31B”, “NZF”) 2: (“2A4”, “RAK”) → (“C7B”, “FAF”) → (“1A2”, “MNY”) Function call: remove(hashTable, key=”C7B”)	0: (“13D”, “ZMD”) → (“10A”, “ABW”) 1: (“31B”, “NZF”) 2: (“2A4”, “RAK”) → (“1A2”, “MNY”)

Question – 5: CO2 [7+8 Points]

I. Given the **array representation** of a binary tree: [None value means the node is empty]
[None, 19, 22, 10, 32, 20, None, 30, None, 5, None, 7, None, None, 2, -2]

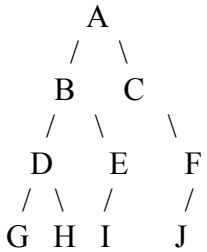
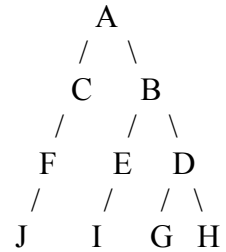
a. **Draw** the binary tree. **[2 marks]**
b. **Write** the post-order and pre-order traversal sequence of the tree. **[1 marks]**
c. Use the **pre-order** traversal sequence in part b to **insert** the elements in that order in an initially empty binary search tree, and show the resulting binary search tree. Note: Consider the first element of the pre-order sequence as the root. **[2 marks]**
d. Perform the following operations **step by step sequentially** on the Binary Search Tree you created in part c. **[2 marks]**
i) Delete node 5 with the help of its successor.
ii) Delete node 20 with the help of its predecessor.

II. Write a **recursive** function **swap_child()** that takes the root of a binary tree, node’s level and a number M as a parameter. The function will swap the left and right children of all the nodes at level M and above. Here, 0 < M < height of the tree (root’s height). Consider, the Node class for Binary Tree already defined with elem, left and right variables. **YOU CANNOT USE LIST OR DICTIONARY, any built-in function, global variables.**

Python Notation:

```
def swap_child(root, level, M):
        # To do
```

Function Call :
swap_child(root, 0, 2). Here root refers to the tree below.

Input Tree	Resulting Tree	Explanation
		Here M = 2 and all the nodes from level 2 and above are swapped left with right. Here above means the level that situated at a higher position of the tree