BRAC UNIVERSITY
Department of Computer Science and Engineering

A

Examination: Final
Duration: 1 Hour 45 Minutes
Number of Questions: 5

CSE220: Data Structures

Semester: Summer 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**.

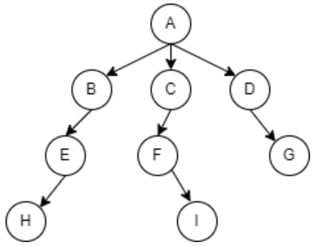
Question 1: CO1 [5 Points]
Write the correct answer in your answer script:

I. What is the output of the following code?

```
def recFunc(x):
   if x<=2:
      return 0
   return  x+recFunc(x//2)
print(recFunc(16))
```

a. 30, b. 24, c. 28, d. 31

II. What is the height of the following tree?



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

III. Suppose, you have an array,
a = [-3, -2, 1, 1, 1, 2, 4, 4, 9]
You want to implement key-indexed searching and sorting on this array. To do that, you build a key-indexed array (auxiliary array) x. What will be the value of x[4]?

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

IV. You want to create a **sorted array** from a **Binary search tree**. For that, which of the following tree-traversal sequences you should find?

a. Pre-order, **b. In-order,**
c. Post-order, **d. Level-order.**

V.

```
def hash_function(input_string, table_size):
      hash_value = 1
      for char in input_string:
            hash_value *= ord(char)-64
      return hash_value % table_size
```

What will be the calculated hash table index from the given hash function if the function parameters are **table_size = 6** and input_string = 'BRAC'?
(The ASCII value of 'A' is 65, 'R' is 82)

a. 0, b. 4, c. 2, d. 108

Question 2: CO4 [10 Points]

Construct a method **isPrime(num)** which will take a number, num as an input. The method will return True if the given number is prime, otherwise will return False.
You have to recursively check for divisibility of the number with the help of a **recursive helper function** having an extra parameter which will be called inside **isPrime(num)** method.

Sample Input	Sample Output
isPrime(51)	False
isPrime(17)	True

Python Notation:
def isPrime(num):
    // To do
	Java Notation:
	public boolean isPrime(num) {
	    // To do
	}

Question – 3: CO3 [2+8 Points]

You are given the hash function, **h(key) = (key^2 + 3) % 6** for a hash-table of length 6. In this hashing, forward chaining is used for resolving conflict and a 6-length array of singly linked lists is used as the hash-table. In the singly linked list, each node has a next pointer, an Integer key and a string value, for example: (4 (key) , “Rafi” (value)). The hash-table stores this key-value pair.

I. What is the hashed-index of **(9, “ABW”)** 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. 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
```

Sample Input and Output:

Some Key-value pairs:
(2, “Abid”) , (4, “Rafi”), (6, “Karim”), (3, “Chitra”), (8, “Nilu”)

HashTable is given in the following:
0: (3, “Chitra”)
1: (8, “Nilu”) → (4, “Rafi”) → (2, “Abid”)
2: None
3: (6, “Karim”)
4: None
5: None

remove(hashTable, key=4) returns the changed hashTable where (4, “Rafi”) is removed.

New HashTable Output:
0: (3, “Chitra”)
1: (8, “Nilu”) → (2, “Abid”)
2: None
3: (6, “Karim”)
4: None
5: None

remove(hashTable, key=9) returns the same given hashTable since 9 doesn’t exist in the table.

Python Notation	Java Notation

def remove (hashTable, key):
    # To do
	public Node[] remove(Node[] hashTable, key) {
	    // To Do
	}

Question – 4: CO2 [10+5 Points]

I. Given the **array representation** of a binary tree: [null value means the node is empty]

[null, 15, 18, 6, 28, 16, null, 23, null, -1, null, 3, null, null, -2, -6]

a. **Draw** the binary tree. **[3 marks]**

b. **Write** the post-order and pre-order traversal sequence of the tree. **[2 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 on the Binary Search Tree you created in part c. **[3 marks]**

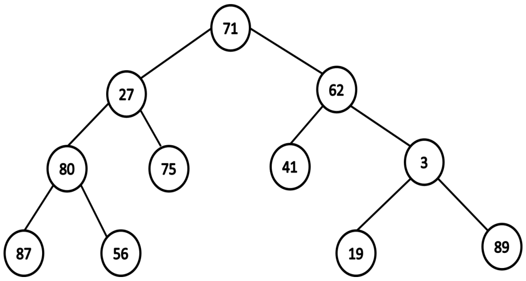
i) Delete node 18 with the help of its successor.
ii) Delete node 15 with the help of its predecessor.

II. Write a **recursive** function **subtract_summation**() that takes the root of a binary tree as a parameter. The function will **subtract** the **summation** of the **right subtree** of the given root **from** the **summation** of the **left subtree** of the given root. Consider, the **Node** class for Binary Tree already defined with elem, left and right variables. You can use helper functions.
YOU CANNOT USE LIST OR DICTIONARY, any built-in function, global variables.

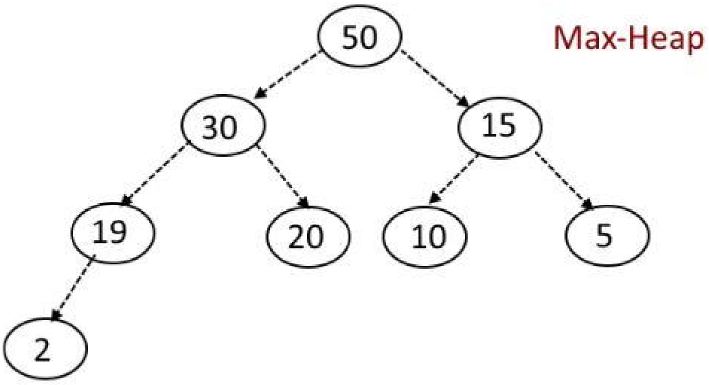
Python Notation:

```
                    def subtract_summation(root):
                              // To do
                              return None
```

Function Call :
print(**subtract_summation**(root)). Here root refers to the tree below.

Sample Input	Sample Output	Explanation
 71 27 62 80 75 41 3 87 56 19 89	111	Summation of left subtree - summation of right subtree = (27+75+80+87+56) - (62+41+3+19+89) = 111

Question – 5: CO2 [5 Points]



The diagram presented above illustrates the tree representation of a specific instance of **Max-heap**. Subsequently, you are required to perform the following operations step by step on this **existing heap**. After each operation, show tree representation of the heap only.

a. insert(75)
b. delete()