BRAC UNIVERSITY
Department of Computer Science and Engineering

A

Examination: Final
Duration: 100 Minutes
Number of Questions: 4

CSE220: Data Structures

Semester: Summer 2024
Full Marks: 40
No. of Pages: 5

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

✔ **Answer all questions. No washroom breaks. Understanding questions is part of the exam.**
✔ At the end of the exam, put the question **paper** inside the answer script and **return both**.

Question 1: CO1 [10 Points]

Write the correct answer in your answer script:

i) Consider the following recursive function:

```
def f(x):
  if x == 1:
     print(x, end=' ')
     return
  f(x // 2)
  print(x, end=' ')
```

f(16)

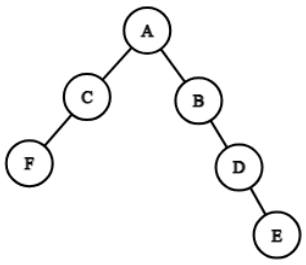
Which of the following is the correct output of the given code?
a) 1 2 4 8 16 **b) 16 8 4 2 1**
c) 1 16 8 4 2 **d) 16 8 4 2 1 2 4 8 16**

ii) In a balanced Binary Search Tree (BST), what happens when a new node is inserted?

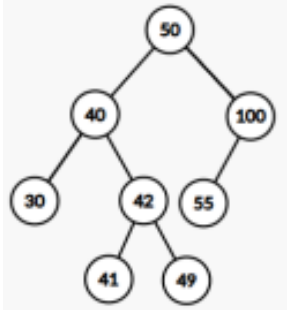
a) The tree stays balanced without any adjustments.
b) The tree's height might increase, and rebalancing may be needed if it becomes unbalanced.
c) The tree will always become unbalanced after the insertion of a new node.
d) The new node is added as a leaf, which keeps the tree balanced by default.

iii) Consider A to be the root of the following tree. What is the order of nodes when the tree is traversed in post-order?

a) ACFBDF **b)** FCEDBA
c) FCABDE **d)** None of the above



iv) Suppose you inserted a node with value 45 in the given BST. Does this operation make this BST unbalanced? What would be the root of the BST if you want to make it balanced?



a) BST will be balanced after the insertion
b) 45 **c)** 42 **d)** 50

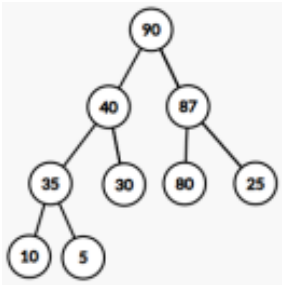
v) Which of the following scenarios requires shifting elements in an array?

a) Inserting an element at a specific position in the array.
b) Accessing an element at a given index.
c) Inserting an element to the end of the array.
d) Updating the value of an element at a specific index.

vi) Suppose, you are given the array representation of a perfect binary tree of height 4. If the index of the leftmost leaf is 16, what is the index of the rightmost leaf?

a) 17
b) 31
c) 4
d) Can't be determined using the provided information

vii) Consider the following heap

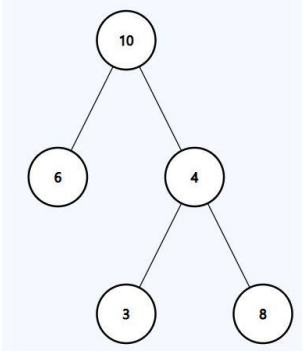


Suppose you are doing a heap sort based on the given heap. After the first step, which node will be the new root?

a) Node with value 40
b) Node with value 87
c) Node with value 80
d) Node with value 10

viii) Consider the following recursive function:

```
def fun(root):
    if root == None:
        return 7
    elif root.elem > 6:
        return fun(root.right) - root.elem
    else:
        return root.elem - fun(root.left)
```



Which of the following is the correct output of the given code for the given binary tree?

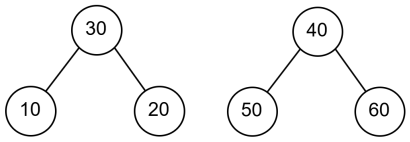
a) 7 b) 2 c) -2 d) 11

ix) Suppose, you are given the array representation of a binary tree:
[None, 6, 8, 4, 3, None, 11, 20, None, 2, None, None, 7, 5]

Which of the following statements is **NOT** correct?

a) Node 20 is the parent of Node 7.
b) Node 11 is the parent of Node 5.
c) This binary tree is not balanced.
d) This binary tree is not full/strict.

x) Consider the following heaps:



Take out one element at a time from each heap (first max heap then min heap) and add it to the new Merged Min Heap. How many total number of swaps you have to make to maintain the min property of the Merged Min Heap.
a) 4 b) 3 c) 5 d) 6

Question 2: CO5 [2+8 Points]

i) Write one advantage and one disadvantage of linked lists over arrays.

ii) Suppose, you are in charge of collecting customer tokens in a line and have to store them in a singly linked list in a first-come, first-served order. Your company's policy is to serve senior citizens first, but you noticed they were standing towards the back, and you know their starting positions. Your task is to rearrange the tokens in a way so that senior citizens are moved to the front.
Implement the method named **rearrange_Tokens(head, seniorPos)**, which takes the head of the linked list and the starting position of the senior citizens. It rearranges senior citizens to the front and others to the back and returns the rearranged linked list's head. If the senior position is invalid, return the original list unchanged.

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

Sample Input	Sample Output	Explanation
Tokens list: A3 → A9 → A4 → A2 → A7 → A8 → A1 Senior citizen position: 4	Rearranged Tokens list: A2 → A7 → A8 → A1 → A3 → A9 → A4	Here the senior citizens start from position 4, hence all the tokens from position 4 to the end of the list were brought to the front and others are behind that list.
Tokens list: A9 → A3 → A4 → A8 → A6 → A5 Senior citizen position: 5	Rearranged Tokens list: A6 → A5 → A9 → A3 → A4 → A8	Here the senior citizens start from position 5, hence all the tokens from position 5 to the end of the list were brought to the front and others are behind that list.

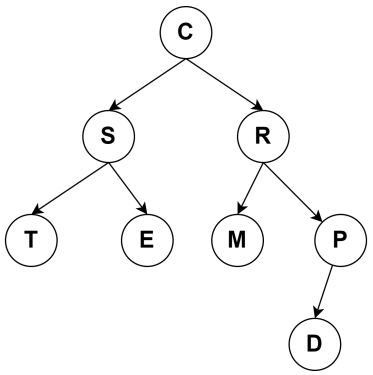
Python Notation	Java Notation

def rearrange_Tokens(head, seniorPos):
  # To Do
	public Node rearrange_Tokens(Node head, int seniorPos) {
	  // To Do
	}

Question 3: CO4 [10 Points]

You are given a binary tree where each node stores a character. You are given the root of a binary tree and a key as inputs. The key is valid if there is a path from the root to a leaf node in the given tree. Now, your Task is to implement a **recursive function** named **is_Valid(root, key)** that determines if the key is valid or not. The function should return True if the key is valid and False otherwise.

Consider the Node class for Binary Tree already defined with elem, left and right variables. You can use helper functions and extra parameters if required. You can not use any other data structure than the binary tree.

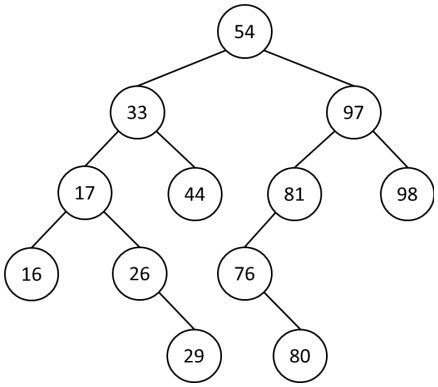
Sample Input	Sample Output	Explanation
	Key: CSE Output: True Key: CRA Output: False Key: CRP Output: False	For CSE there is a path from root to a leaf node where adding the characters of the nodes makes CSE. So the key is Valid. For CRA & CRP, there is no such path from root to a leaf node that creates the key. So they are invalid. For CRP the Node P is not leaf node hence it is ignored.

Python Notation	Java Notation

def is_Valid(root, key):
  # To Do
	public boolean is_Valid(Node root, String key) {
	  // To Do
	}

Question 4: CO2 [10 Points]

Take a look at the following binary search tree. Do the following operations step by step:



a) Traverse the given tree in pre-order and write the output in your script. [2]
b) Insert the key 89 in the given BST and draw the BST. [3]
c) Is the modified tree a balanced binary tree? (Yes/No) [2]
d) We know that the in-order traversal of BST provides an ascending order of the elements. Can you propose an order of traversal that will print the BST in descending order without using any other data structure? Apply it and print the elements. [3]

Question 5: Bonus [5 Points]

Complete the recursive function print_all_directories, which will take the root directory and recursively print all the directory structures. Analyze the given code below to understand how it works. **You only need to write the print_all_directories function in your script.**

```
class DirectoryNode:
    def __init__(self, name, is_folder, current=None, next=None):
        self.name = name  # Name of the folder or file
        self.is_folder = is_folder  # True if it's a folder, False if it's a file
        self.current = current  # Points to the first child (for folders)
        self.next = next  # Points to the next sibling (folder or file)

# Function to print all directories and files in a structured manner
def print_all_directories(root, level=0):
    # To Do

# Driver Code
# Sample directory structure
root = DirectoryNode("root", True)

# Files in the root folder
root.current = DirectoryNode("book.txt", False)

# Anime folder inside root
anime_folder = DirectoryNode("Anime", True)
anime_folder.current = DirectoryNode("Attack On Titan", True)
anime_folder.current.current = DirectoryNode("AOTS1E1.mkv", False)
anime_folder.current.current.next = DirectoryNode("AOTS1E2.mkv", False)

# Connect Anime folder as the sibling of movie.mkv
root.current.next = anime_folder

# Movie folder inside root
movie_folder = DirectoryNode("Movie", True)
movie_folder.current = DirectoryNode("Captain America.mkv", False)

# Connect the Movie folder as the sibling of the Anime folder
root.current.next.next = movie_folder

# Print the directory structure
print_all_directories(root)
```

Sample Output:

```
root (Folder)
        book.txt (File)
        Anime (Folder)
                Attack On Titan (Folder)
                        AOTS1E1.mkv (File)
                        AOTS1E2.mkv (File)
        Movie (Folder)
                Captain America.mkv (File)
```

Hint: The indentations represent files' and folders' parent directories. You can use "\t" for printing the indentation. 2 * "\t" means 2 tabs before printing. The level parameter maintains this.