DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY,
CSJM UNIVERSITY, KANPUR

Subject - Data Structure Using Python (CSE-S208) Branch CSE-AI

Semester- 2023-2024 (Even Semester) Year: **II Year, Semester IV**

Second Mid Semester Examination, 13/02/2024, Shift-II

Time: 1:30 hours Maximum Marks: 30

Note: All questions are compulsory.

Section A

Each question in this section carries 01 marks

1. Which data structure is used extensively in a syntax analyzer to check matching parenthesis?

   ANS: Stack

2. How many minimum number of queues are required to implement a stack using queues?

   ANS: 2

3. How many pointers need to be updated to insert a new node in the middle of a double-linked list?

   ANS: 2

4. What would be the top of the stack after the evaluation of below expression?

   6 2 3 + - 3 8 2 / + * 3 ^ 2 +

   ANS: 345

5. How many pointers are required to build a double-linked list of length **N**? (Ignore the None pointers)

   ANS: 2N-2

6. What should be the overflow condition of a circular queue of size **S**, with rear and front pointers **R** and **F**, respectively?

   ANS: (R+1)%S == F

7. What is the maximum number of leaf nodes possible for a binary tree of height **10**?

   ANS: 2^{10}

8. What is the worst-case time complexity of search operation in a Binary Search Tree (BST)?

   ANS: O(n)

9. What is the efficient data structure for implementing a sparse Binary Tree (BT)?

   ANS: Double linked list

Section B

Each question in this section carries 03 marks

10. Assume you are given with a pointer to the head of a single linked list **h**. Write a function (`def reverse_list(h: Node) -> None:`) in Python to reverse the links of the linked list **h**. (*Function must work for all possible inputs*)

ANS:

```
def reverse_list(h: Node) -> None:
    p: Node | None = None
    q: Node | None = None
    r = h
    while r:
        p = r.next
        r.next = q
        q = r
        r = p
    h = q
```

11. Convert the following infix expression into a prefix and postfix. What would be the required size of the stack to evaluate the postfix? (1+1+1)

(((1 + 2) * (3 + 4)) / 3) + (6 * (7 + 8))

ANS:

(i) Prefix: + / * + 1 2 + 3 4 3 * 6 + 7 8

(ii) Postfix: 1 2 + 3 4 + * 3 / 6 7 8 + * +

(iii) Stack size: 4

12. A pointer was given to the head of a double-linked list **h**. Write a function (`def length(h: Node) -> int:`) that recursively computes the list's length. (*Function must work for all possible inputs*)

ANS:

```
def length(h: Node) -> int:
    if h:
        return 1 + length(h.next)
    return 0
```

Section C

Each question in this section carries 06 marks

13. Considers the following Node in Python below for Binary Search Tree (BST):

```
class Node:
    def __init__(self, data: any) -> None:
        self.data: any = data
        self.left: Node | None = None
        self.right: Node | None = None
```

Given the root node **R** of a BST, implement following three functions based on the given Node in Python to print: (2+2+2)

(a) In order traversal: `def in_order(R: Node) -> None`

ANS:

```
def in_order(R: Node) -> None:
    if R:
        in_order(R.left)
        print(R.data)
        in_order(R.right)
```

(b) Pre order traversal: `def pre_order(R: Node) -> None`

ANS:

```
def pre_order(R: Node) -> None:
    if R:
        print(R.data)
        pre_order(R.left)
        pre_order(R.right)
```

(c) Post order traversal: `def post_order(R: Node) -> None`

ANS:

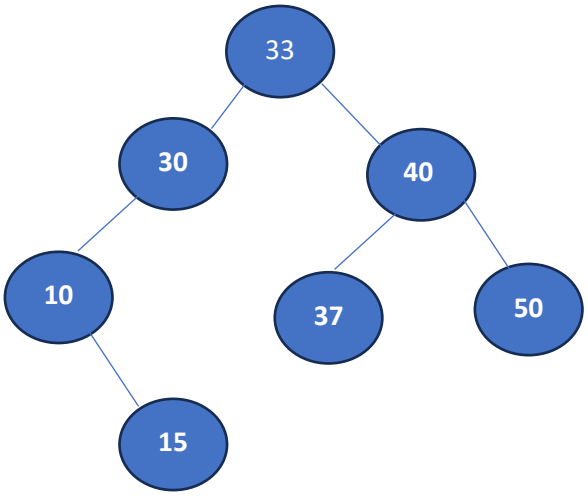
```
def post_order(R: Node) -> None:
    if R:
        post_order(R.left)
        post_order(R.right)
        print(R.data)
```

Note: *Functions must work for all possible inputs*

14. Considering the following two insertion sequence in the Binary Search Tree (BST), first build the BST and then, mention which BST is better and why? (2+2+2)
(a) 33, 40, 30, 10, 15, 50, 37
(b) 5, 1, 6, 7, 10, 9, 8

ANS:

(a) Better because of height is almost balanced hence less search complexity.



(b) High complexity due to the skewed structure.