HW 8

Design & Analysis of Algorithms

Pranav Umakant Pujar

1001965075

10.1-1) The operations can be represented as:-

PUSH(S,4) : S= 4 _ _ _ _ _

PUSH(S,1) : S= 4 1 _ _ _ _

PUSH(S,3) : S= 4 1 3 _ _ _

POP(S) : S= 4 1 ⋮ 3 _ _ _

S.top=1 ↗ , ↖ popped

PUSH(S,8) : S= 4 1 8 ⋮ 3 _ _

PUSH : S= 4 1 ⋮ 8 3 _ _

S.top=1 ↗ popped elements

10.1-3) The operations can be represented as:-

ENQUEUE(Q,4) : Q= _ _ _ ⋮ 4 _ _

Using the wrap-around feature of array-based queue. The queue starts from the right of the boundary (dotted line)

ENQUEUE(Q,1) : Q= _ _ _ ⋮ 4 1 _

ENQUEUE(Q,3) : Q= _ _ _ ⋮ 4 1 3

DEQUEUE(Q) : Q= _ _ _ 4 ⋮ 1 3

Before dequeue operation	After dequeue
Q.tail = 1	Q.tail = 1
Q.head = 4	Q.head = 5

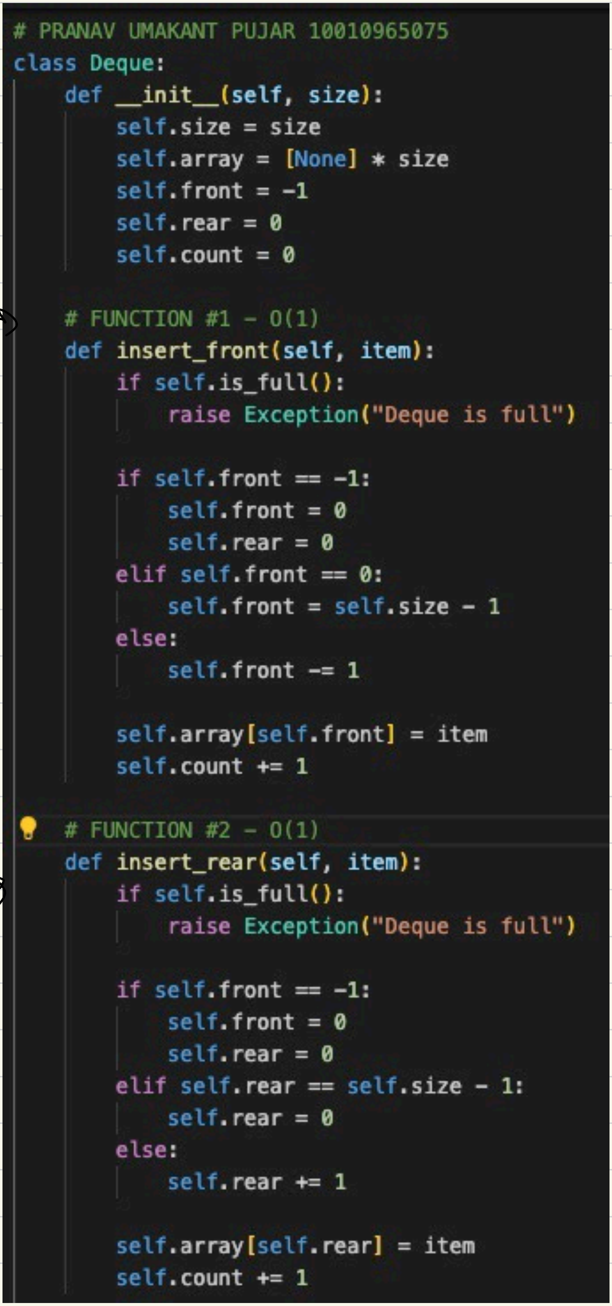
ENQUEUE (Q,8) : Q= 8 ⋮ _ _ 4 ⋮ 1 3

Q.head = 5
Q.tail = 2

DEQUEUE (Q) : Q= 8 ⋮ _ _ 4 1 ⋮ 3

Q.head = 6
Q.tail = 2

(0.1-5)



```python
# PRANAV UMAKANT PUJAR 10010965075
class Deque:
    def __init__(self, size):
        self.size = size
        self.array = [None] * size
        self.front = -1
        self.rear = 0
        self.count = 0
```

function #1

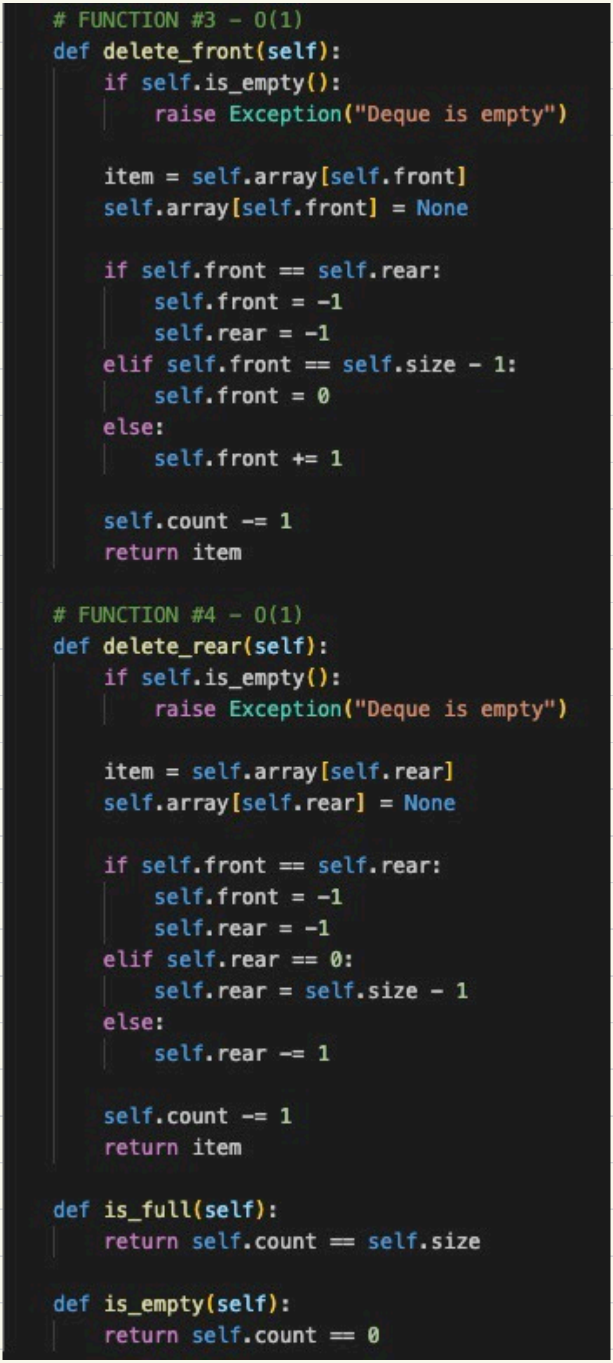
```python
    # FUNCTION #1 - O(1)
    def insert_front(self, item):
        if self.is_full():
            raise Exception("Deque is full")

        if self.front == -1:
            self.front = 0
            self.rear = 0
        elif self.front == 0:
            self.front = self.size - 1
        else:
            self.front -= 1

        self.array[self.front] = item
        self.count += 1
```

function #2

```python
    # FUNCTION #2 - O(1)
    def insert_rear(self, item):
        if self.is_full():
            raise Exception("Deque is full")

        if self.front == -1:
            self.front = 0
            self.rear = 0
        elif self.rear == self.size - 1:
            self.rear = 0
        else:
            self.rear += 1

        self.array[self.rear] = item
        self.count += 1
```

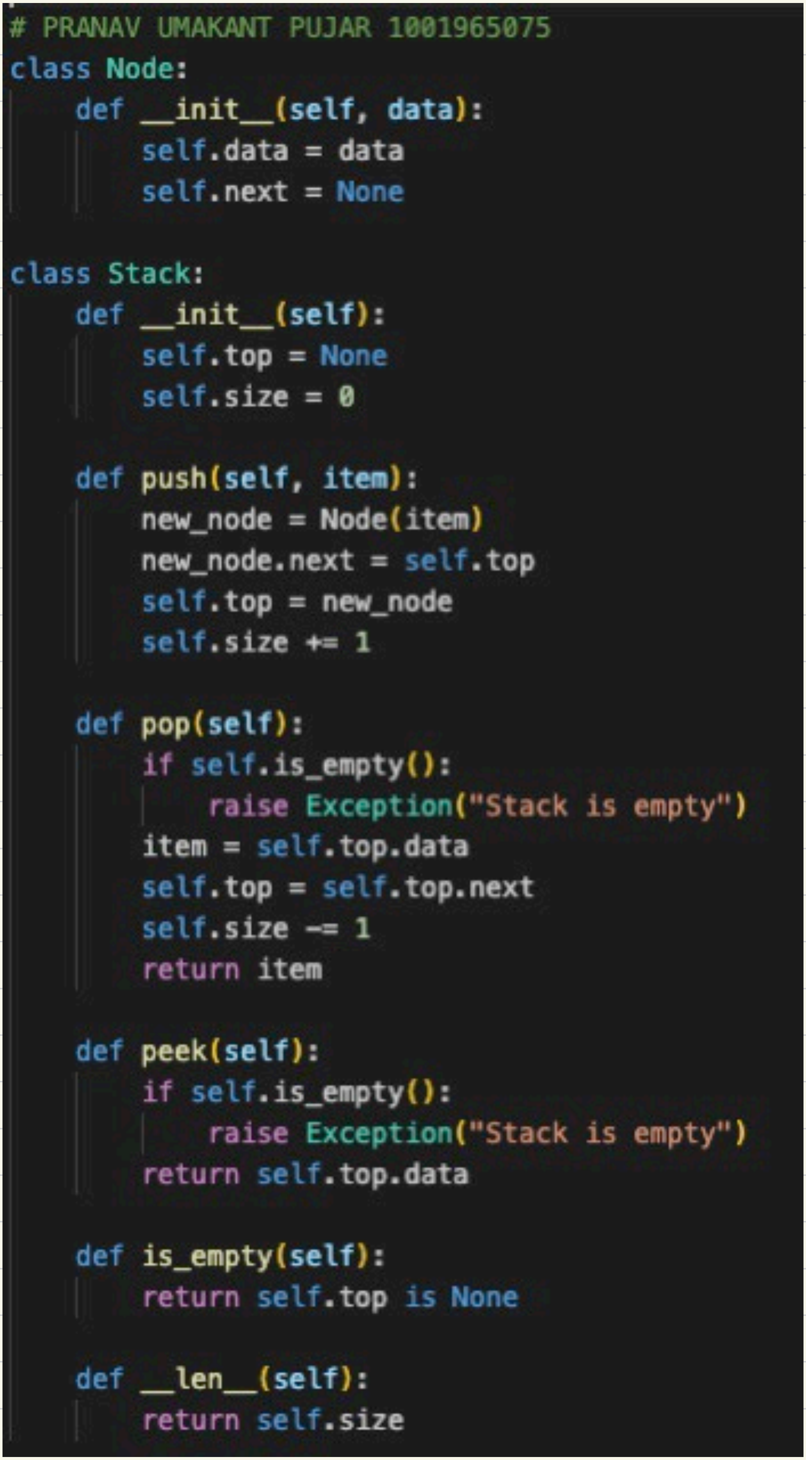
function #3 →

function #4 →

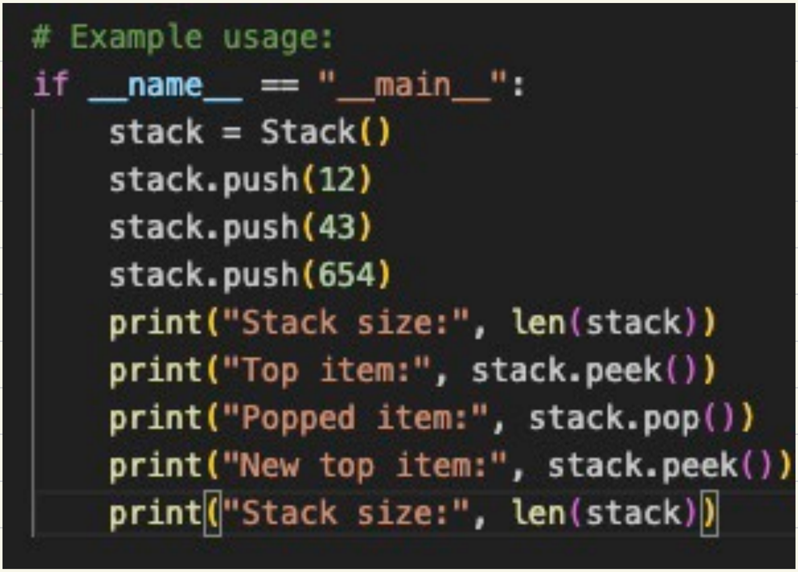


```python
# FUNCTION #3 - O(1)
def delete_front(self):
    if self.is_empty():
        raise Exception("Deque is empty")

    item = self.array[self.front]
    self.array[self.front] = None

    if self.front == self.rear:
        self.front = -1
        self.rear = -1
    elif self.front == self.size - 1:
        self.front = 0
    else:
        self.front += 1

    self.count -= 1
    return item

# FUNCTION #4 - O(1)
def delete_rear(self):
    if self.is_empty():
        raise Exception("Deque is empty")

    item = self.array[self.rear]
    self.array[self.rear] = None

    if self.front == self.rear:
        self.front = -1
        self.rear = -1
    elif self.rear == 0:
        self.rear = self.size - 1
    else:
        self.rear -= 1

    self.count -= 1
    return item

def is_full(self):
    return self.count == self.size

def is_empty(self):
    return self.count == 0
```

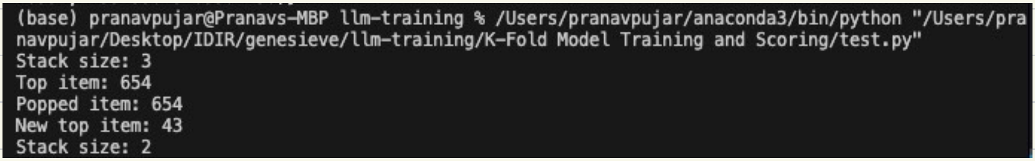
10.2-2)



```python
# PRANAV UMAKANT PUJAR 1001965075
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class Stack:
    def __init__(self):
        self.top = None
        self.size = 0

    def push(self, item):
        new_node = Node(item)
        new_node.next = self.top
        self.top = new_node
        self.size += 1

    def pop(self):
        if self.is_empty():
            raise Exception("Stack is empty")
        item = self.top.data
        self.top = self.top.next
        self.size -= 1
        return item

    def peek(self):
        if self.is_empty():
            raise Exception("Stack is empty")
        return self.top.data

    def is_empty(self):
        return self.top is None

    def __len__(self):
        return self.size
```



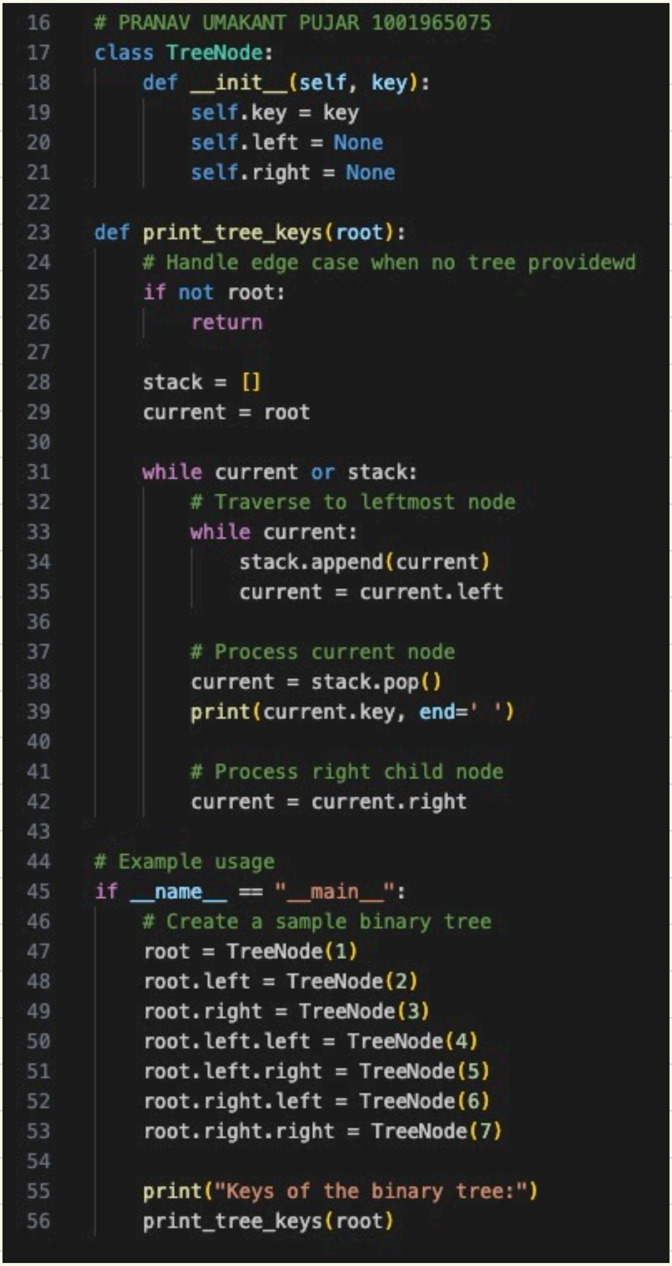
```
# Example usage:
if __name__ == "__main__":
    stack = Stack()
    stack.push(12)
    stack.push(43)
    stack.push(654)
    print("Stack size:", len(stack))
    print("Top item:", stack.peek())
    print("Popped item:", stack.pop())
    print("New top item:", stack.peek())
    print("Stack size:", len(stack))
```



```
(base) pranavpujar@Pranavs-MBP llm-training % /Users/pranavpujar/anaconda3/bin/python "/Users/pra
navpujar/Desktop/IDIR/genesieve/llm-training/K-Fold Model Training and Scoring/test.py"
Stack size: 3
Top item: 654
Popped item: 654
New top item: 43
Stack size: 2
```

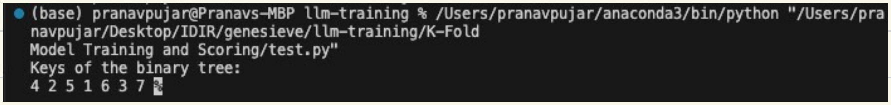
Output of Example Usage ↑

10-4-3)



```python
# PRANAV UMAKANT PUJAR 1001965075
class TreeNode:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

def print_tree_keys(root):
    # Handle edge case when no tree providewd
    if not root:
        return

    stack = []
    current = root

    while current or stack:
        # Traverse to leftmost node
        while current:
            stack.append(current)
            current = current.left

        # Process current node
        current = stack.pop()
        print(current.key, end=' ')

        # Process right child node
        current = current.right

# Example usage
if __name__ == "__main__":
    # Create a sample binary tree
    root = TreeNode(1)
    root.left = TreeNode(2)
    root.right = TreeNode(3)
    root.left.left = TreeNode(4)
    root.left.right = TreeNode(5)
    root.right.left = TreeNode(6)
    root.right.right = TreeNode(7)

    print("Keys of the binary tree:")
    print_tree_keys(root)
```

Output:



```
(base) pranavpujar@Pranavs-MBP llm-training % /Users/pranavpujar/anaconda3/bin/python "/Users/pra
navpujar/Desktop/IDIR/genesieve/llm-training/K-Fold
Model Training and Scoring/test.py"
Keys of the binary tree:
4 2 5 1 6 3 7 %
```