Name: Hemang Ranga

Roll no: 20BCS057

OOP Task-4

1

```
#class Node
class Node:
  def __init__(self, data):
    self.data = data
    self.next = None # address of next node is NULL(None) as default

class Stack:
  def __init__(self):
    self.head = None #default value of head is NULL(None)

  #function to check stack is empty or not
  def isempty(self):
    if self.head == None:
      return True
    else:
      return False

  #function to add element to the stack
  def add(self, d):
    if self.head == None:
      self.head = Node(d)
    else:
      newnode = Node(d)
      newnode.next = self.head
      self.head = newnode

  #function to delete data
  def remove(self):
    if self.isempty():
      return None
    else:
      removed = self.head
      self.head = self.head.next
      removed.next = None
      return removed.data
  #function to print element of stack
  def print_stack(self):
    var_node = self.head
    if self.isempty():
      print("Empty Stack")
    else:
      while(var_node != None):
        print(var_node.data,"->",end = " ")
        var_node = var_node.next
      return
  def __del__(self):

    print("\n\nEnd of program.")
    temp = self.head
    while temp is not None:
      temp1 = temp.next
      del temp
      temp = temp1
    return

# Main code
MyStack = Stack()

# (a) Check stack is empty
x = MyStack.isempty()
if x == True:
  print("Empty")
else:
  print("not Empty")

# (b) Add data to stack
MyStack.add(5)
MyStack.add(10)
MyStack.add(15)
MyStack.add(20)

# (d) Display the elements of the stack.
print("\n")
MyStack.print_stack()

# (c) Delete top elements of stack
MyStack.remove()
MyStack.remove()

# (d) Display the elements of the stack.
print("\n")
MyStack.print_stack()

# (e) Deallocate the memory assigned to each node using destructors.
del MyStack
```

```
Empty


20 -> 15 -> 10 -> 5 ->

10 -> 5 ->

End of program.
```

2

```python
# default constructor
class Rectangle:
  def __init__(self):
    self.length = 8
    self.breadth = 6
  def area(self):
    a = (self.length) * (self.breadth)
    return a

rect = Rectangle()
print(rect.area())
```

```
48
```

```python
# parametrized constructor
class Rectangle:
  def __init__(self, l, b):
    self.length = l
    self.breadth = b
  def area(self):
    a = (self.length) * (self.breadth)
    return a

rect = Rectangle(7,9)
print(rect.area())
```

```
63
```

3

```python
class Queue:
    def __init__(self, size):
        self.queue = []
        self.size = size

    def   str  (self):
        myString = ' '.join(str(i) for i in self.queue)
        return myString

    def enqueue(self, item):
        #This function adds an item to the rear end of the queue
        if(self.isFull() != True):
            self.queue.insert(0, item)
        else:
            print('Queue is Full!')

    def dequeue(self):
        #This function removes an item from the front end of the queue
        if(self.isEmpty() != True):
            return self.queue.pop()
        else:
            print('Queue is Empty!')

    def isEmpty(self):
        #This function checks if the queue is empty
        return self.queue == []

    def isFull(self):
        #This function checks if the queue is full
        return len(self.queue) == self.size

    def peek(self):
        #This function helps to see the first element at the fron end of the queue
        if(self.isEmpty() != True):
            return self.queue[-1]
        else:
            print('Queue is Empty!')


if __name__ == '__main__':
    myQueue = Queue(10)
    myQueue.enqueue(7)
    myQueue.enqueue(8)
    myQueue.enqueue(9)

    print(myQueue)

    myQueue.enqueue(17)
    myQueue.enqueue(2)
    myQueue.enqueue(13)

    print(myQueue)

    myQueue.dequeue()

    print(myQueue)
```

```
9 8 7
13 2 17 9 8 7
13 2 17 9 8
```

✓ 0s completed at 4:56 PM ● ×