

Name- Harshit Kumar
Subject- Data Structure Using python
Reg.no- 21ETMC412011
Section- Mathematics and Computing (M&C)
Department- Cse

Total Marks: 12
31/03/2023

Submission Date:

Assignment Problems(2 marks each)

1. Implement the stack Data Structure using a fixed capacity array (underlying storage).
2. Implement the singly linked list (instance of Singly list keep reference to head, tail and size).
3. Implement Deque data structure(time complexity of each method must be $O(1)$) using array.
4. Implement Deque data structure(time complexity of each method must be $O(1)$) using doubly linked list.
5. Implement stack data structure using two queues.
6. Implement queue data structure using two stacks.

Note: Paste the screen-shots of code and output of each problem

P.T.O

[Question-1]

```
main.py > ...
1 class Stack:
2     def __init__(self, size=10):
3         self._stack = []
4         self._size = size
5
6     def is_empty(self):
7         return len(self._stack) <= 0
8
9     def push(self, data):
10        if len(self._stack) >= self._size:
11            raise Exception('Stack overflow')
12        else:
13            self._stack.append(data)
14
15    def pop(self):
16        if len(self._stack) <= 0:
17            raise Exception('Stack underflow')
18        else:
19            return self._stack.pop()
20
21    def peek(self):
22        if len(self._stack) <= 0:
23            raise Exception('Stack underflow')
24        else:
25            return self._stack[-1]
26
27    def length(self):
28        return len(self._stack)
29
30 stack = Stack(5)
31 stack.push(5)
32 stack.push(2)
33 stack.push(3)
34 stack.push(9)
35 stack.push(6)
36 print(stack.peek())
37 print(stack.pop())
38 print(stack.length())
39 print(stack.peek())
```

Output-

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
	6		
	6		
	4		
	9		
	PS C:\Users\suruc\PycharmProjects\firstfrog> █		

[Question-2]

```
main.py > SinglyLinkedList
1 class ListNode:
2     def __init__(self, val=None):
3         self.val = val
4         self.next = None
5
6 class SinglyLinkedList:
7     def __init__(self):
8         self.head = None
9         self.tail = None
10        self.size = 0
11
12    def is_empty(self):
13        return self.size == 0
14
15    def append(self, val):
16        node = ListNode(val)
17        if self.is_empty():
18            self.head = node
19        else:
20            self.tail.next = node
21            self.tail = node
22            self.size += 1
23
24    def prepend(self, val):
25        node = ListNode(val)
26        if self.is_empty():
27            self.tail = node
28        else:
29            node.next = self.head
30            self.head = node
31            self.size += 1
32
33    def insert_after(self, prev_node, val):
34        node = ListNode(val)
35        node.next = prev_node.next
36        prev_node.next = node
37        if prev_node == self.tail:
38            self.tail = node
39            self.size += 1
40
41    def delete(self, val):
42        if self.is_empty():
43            raise Exception('List is empty')
44        if self.head.val == val:
45            self.head = self.head.next
46            if self.head is None:
47                self.tail = None
48            self.size -= 1
49            return
50        node = self.head
51        while node.next is not None:
52            if node.next.val == val:
53                if node.next == self.tail:
54                    self.tail = node
55                node.next = node.next.next
56                self.size -= 1
57                return
58            node = node.next
59        raise Exception('Value not found in list')
60 linked_list = SinglyLinkedList()
61
62 linked_list.append(1)
63 linked_list.append(2)
64 linked_list.prepend(0)
65 linked_list.insert_after(linked_list.head.next, 1.5)
66 linked_list.delete(0)
67 linked_list.delete(2)
68
69 print('List size:', linked_list.size)
70 node = linked_list.head
71 while node is not None:
72     print(node.val)
73     node = node.next
```

Output-

```
List size: 2
1
1.5
PS C:\Users\suruc\PycharmProjects\firstfrog> █
```

[Question-3]

```
main.py > ...
1 class Deque:
2     def __init__(self, capacity):
3         self.capacity = capacity
4         self.items = [None] * capacity
5         self.front = 0
6         self.rear = 0
7         self.size = 0
8
9     def is_empty(self):
10        return self.size == 0
11
12    def is_full(self):
13        return self.size == self.capacity
14
15    def push_front(self, item):
16        if self.is_full():
17            raise Exception('Deque is full')
18        self.front = (self.front - 1 + self.capacity) % self.capacity
19        self.items[self.front] = item
20        self.size += 1
21
22    def push_rear(self, item):
23        if self.is_full():
24            raise Exception('Deque is full')
25        self.items[self.rear] = item
26        self.rear = (self.rear + 1) % self.capacity
27        self.size += 1
28
29    def pop_front(self):
30        if self.is_empty():
31            raise Exception('Deque is empty')
32        item = self.items[self.front]
33        self.front = (self.front + 1) % self.capacity
34        self.size -= 1
35        return item
36
37    def pop_rear(self):
38        if self.is_empty():
39            raise Exception('Deque is empty')
40        self.rear = (self.rear - 1 + self.capacity) % self.capacity
41        item = self.items[self.rear]
42        self.size -= 1
43        return item
```

```
44 deque = Deque(5)
45
46 deque.push_front(1)
47 deque.push_front(0)
48 deque.push_rear(2)
49 deque.push_rear(3)
50 deque.pop_front()
51 deque.pop_rear()
52
53 print('Deque size:', deque.size)
54 while not deque.is_empty():
55     print(deque.pop_front())
```

Output-

```
Deque size: 2
1
2
PS C:\Users\suruc\PycharmProjects\firstfrog> █
```

[Question-4]

```
main.py > ...
1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.next = None
5         self.prev = None
6
7 class Deque:
8     def __init__(self):
9         self.front = None
10        self.rear = None
11
12    def is_empty(self):
13        return self.front is None
14
15    def add_front(self, data):
16        new_node = Node(data)
17        if self.is_empty():
18            self.front = self.rear = new_node
19        else:
20            new_node.next = self.front
21            self.front.prev = new_node
22            self.front = new_node
23
24    def add_rear(self, data):
25        new_node = Node(data)
26        if self.is_empty():
27            self.front = self.rear = new_node
28        else:
29            new_node.prev = self.rear
30            self.rear.next = new_node
31            self.rear = new_node
32
33    def remove_front(self):
34        if self.is_empty():
35            return None
36        front_data = self.front.data
37        if self.front == self.rear:
38            self.front = self.rear = None
39        else:
40            self.front = self.front.next
41            self.front.prev = None
42        return front_data
43
44    def remove_rear(self):
45        if self.is_empty():
46            return None
47        rear_data = self.rear.data
48        if self.front == self.rear:
49            self.front = self.rear = None
50        else:
51            self.rear = self.rear.prev
52            self.rear.next = None
53        return rear_data
54
```

```
55    def peek_front(self):
56        return self.front.data if self.front else None
57
58    def peek_rear(self):
59        return self.rear.data if self.rear else None
60
61    dq = Deque()
62    dq.add_front(1)
63    dq.add_front(2)
64    dq.add_rear(3)
65    dq.add_rear(4)
66    print(dq.peek_front())
67    print(dq.peek_rear())
68    print(dq.remove_front())
69    print(dq.remove_rear())
70    print(dq.peek_front())
71    print(dq.peek_rear())
```

Output-

```
2
4
1
3
PS C:\Users\suruc\PycharmProjects\firstfrog> █
```

[Question-5]

```
main.py > ...
1  class Stack:
2      def __init__(self):
3          self.queue1 = []
4          self.queue2 = []
5
6      def push(self, item):
7          self.queue1.append(item)
8
9      def pop(self):
10         if not self.queue1:
11             return None
12
13         while len(self.queue1) > 1:
14             self.queue2.append(self.queue1.pop(0))
15
16         popped_item = self.queue1.pop(0)
17
18         self.queue1, self.queue2 = self.queue2, self.queue1
19
20         return popped_item
21
22     def peek(self):
23         if not self.queue1:
24             return None
25
26         while len(self.queue1) > 1:
27             self.queue2.append(self.queue1.pop(0))
28
29         peeked_item = self.queue1[0]
30
31         self.queue2.append(self.queue1.pop(0))
32
33         self.queue1, self.queue2 = self.queue2, self.queue1
34
35         return peeked_item
36
37     def is_empty(self):
38         return not bool(self.queue1)
39
40     def size(self):
41         return len(self.queue1)
42
43
44 s = Stack()
45 s.push(1)
46 s.push(2)
47 s.push(3)
48 print(s)
49 print(s.pop())
50 print(s.peek())
51 print(s.pop())
52 print(s.pop())
53 print(s.is_empty())
```

Output-

```
3
2
2
1
True
PS C:\Users\suruc\PycharmProjects\firstfrog> 
```

[Question-6]

```
main.py > Queue > deQueue
1  class Queue:
2      def __init__(self):
3          self.s1 = []
4          self.s2 = []
5
6
7      def enqueue(self, x):
8          self.s1.append(x)
9
10
11      def dequeue(self):
12
13          if len(self.s1) == 0 and len(self.s2) == 0:
14              print("Q is Empty")
15              return
16
17          elif len(self.s2) == 0 and len(self.s1) > 0:
18              while len(self.s1):
19                  temp = self.s1.pop()
20                  self.s2.append(temp)
21              return self.s2.pop()
22
23          else:
24              return self.s2.pop()
25
26
27  if __name__ == '__main__':
28      q = Queue()
29      q.enqueue(1)
30      q.enqueue(2)
31      q.enqueue(3)
32
33      print(q.dequeue())
34      print(q.dequeue())
35      print(q.dequeue())
```

Output-

```
1
2
3
PS C:\Users\suruc\PycharmProjects\firstfrog> 
```