Experiment No. 6

Program for data structure using built in function for link list, stack and queues
Date of Performance:
Date of Submission:

Experiment No. 6

Title: Program for data structure using built in function for link list, stack and queues

Aim: To study and implement data structure using built in function for link list, stack and queues

Objective: To introduce data structures in python

Theory:

Stacks -the simplest of all data structures, but also the most important. A stack is a collection of objects that are inserted and removed using the LIFO principle. LIFO stands for "Last In First Out". Because of the way stacks are structured, the last item added is the first to be removed, and vice-versa: the first item added is the last to be removed.

Queues – essentially a modified stack. It is a collection of objects that are inserted and removed according to the FIFO (First In First Out) principle. Queues are analogous to a line at the grocery store: people are added to the line from the back, and the first in line is the first that gets checked out – BOOM, FIFO!

Linked Lists

The Stack and Queue representations I just shared with you employ the python-based list to store their elements. A python list is nothing more than a dynamic array, which has some disadvantages.

The length of the dynamic array may be longer than the number of elements it stores, taking up precious free space.

Insertion and deletion from arrays are expensive since you must move the items next to them over

Using Linked Lists to implement a stack and a queue (instead of a dynamic array) solve both of these issues; addition and removal from both of these data structures (when implemented with a linked list) can be accomplished in constant O(1) time. This is a HUGE advantage when dealing with lists of millions of items.

Linked Lists – comprised of 'Nodes'. Each node stores a piece of data and a reference to its next and/or previous node. This builds a linear sequence of nodes. All Linked Lists store a head, which is a reference to the first node. Some Linked Lists also store a tail, a reference to the last node in the list.

```python
#Experiment 6
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def push(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    def pop(self):
        if self.head is None:
            return None
        data = self.head.data
        self.head = self.head.next
        return data

    def print_list(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")

class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        return None

    def peek(self):
        if not self.is_empty():
            return self.items[-1]
        return None

```

```
    def is_empty(self):
        return len(self.items) == 0

    def print_stack(self):
        print("Top -> ", end="")
        for item in self.items[::-1]:
            print(item, end=" -> ")
        print("Bottom")

class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        return None

    def peek(self):
        if not self.is_empty():
            return self.items[0]
        return None

    def is_empty(self):
        return len(self.items) == 0

    def print_queue(self):
        print("Front -> ", end="")
        for item in self.items:
            print(item, end=" -> ")
        print("Rear")

linked_list = LinkedList()
linked_list.push(5)
linked_list.push(3)
linked_list.push(1)
print("Linked List:")
linked_list.print_list()
print("\n")

stack = Stack()
stack.push(4)
stack.push(2)
stack.push(0)
print("Stack:")
stack.print_stack()
print("\n")

queue = Queue()
```

```
Linked List:
1 -> 3 -> 5 -> None


Stack:
Top -> 0 -> 2 -> 4 -> Bottom


Queue:
Front -> 7 -> 6 -> 8 -> Rear


...Program finished with exit code 0
Press ENTER to exit console.
```

(Output)

Conclusion: Data structures python has been studied and implemented.

This program effectively demonstrates linked lists, stacks, and queues using built-in functions, serving as a well-structured and understandable implementation. Considering the suggestions for error handling and exploring custom implementations can further enhance its functionality and efficiency.