Experiment No. 7
Program for data structure using built in function for link list, stack and queues
Date of Performance:
Date of Submission:

Experiment No. 7

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.

Code 1 :

```
# stack using list

stack = ["ABCD", "HIJK", "OPQR"]

stack.append("VWXY")

stack.append("SUVS")

print(stack)


# Removes the last item

print(stack.pop())


print(stack)


# Removes the last item

print(stack.pop())


print(stack)
```

Output 1 :

```
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\>python stack.py
['ABCD', 'HIJK', 'OPQR', 'VWXY', 'SUVS']
SUVS
['ABCD', 'HIJK', 'OPQR', 'VWXY']
VWXY
['ABCD', 'HIJK', 'OPQR']

D:\>
```

Code 2 :

```
# Queue using list

queue = ["Riya", "Jiya", "Chiya"]

queue.append("Maya")

queue.append("Chaya")

print(queue)


# Removes the first item

print(queue.pop(0))
```

```
print(queue)


# Removes the first item

print(queue.pop(0))


print(queue)
```

Output 2 :

```
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\>python queue.py
['Riya', 'Jiya', 'Chiya', 'Maya', 'Chaya']
Riya
['Jiya', 'Chiya', 'Maya', 'Chaya']
Jiya
['Chiya', 'Maya', 'Chaya']

D:\>
```

Code 3 :

```
import collections


# Linked List
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


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


    def append(self, data):
        if not self.head:
            self.head = Node(data)
        else:
            current = self.head
            while current.next:
```

```
            current = current.next
        current.next = Node(data)

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

if __name__ == "__main__":
    print("Linked List:")
    linked_list = LinkedList()
    linked_list.append("first")
    linked_list.append("second")
    linked_list.append("third")
    linked_list.display()
```

Output:

```
Microsoft Windows [Version 10.0.18363.1198]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\>python Linkedlist.py
Linked List:
first -> second -> third -> None

D:\>
```

Conclusion: Data structures python has been studied and implemented.

The program demonstrates the implementation of fundamental data structures - linked lists, stacks, and queues - using Python's built-in functionalities. Linked lists provide a linear structure for storing elements with dynamic memory allocation. Stacks facilitate Last In, First Out (LIFO) access to elements, while queues offer First In, First Out (FIFO) access, showcasing essential concepts in data structure manipulation. This foundational demonstration sets the stage for understanding and implementing more complex data structures and algorithms.