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:

```
def traverseList(lst):
    print("List elements:")
    for item in lst:
        print(item)

def displayList(lst):
    print("List:", lst)

def insertElement(lst, index, element):
    lst.insert(index, element)
    print(f'Element {element} inserted at index {index}")
```

```
def appendElement(lst, element):
    lst.append(element)
    print(f"Element {element} appended to the list")

def removeElement(lst, element):
    if element in lst:
        lst.remove(element)
        print(f"Element {element} removed from the list")
    else:
        print(f"Element {element} not found in the list")

def replaceElement(lst, index, newElement):
    if index < len(lst):
        lst[index] = newElement
        print(f"Element at index {index} replaced with {newElement}")
    else:
        print("Index out of range")

def searchElement(lst, element):
    if element in lst:
        print(f"Element {element} found at index {lst.index(element)}")
    else:
        print(f"Element {element} not found in the list")

def list_size(lst):
    print(f"Size of the list: {len(lst)}")

myList = []
```

```
while True:
    print("\n1. Display list")
    print("2. Traverse list")
    print("3. Insert element")
    print("4. Append element")
    print("5. Remove element")
    print("6. Replace element")
    print("7. Search element")
    print("8. Size of list")
    print("9. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':
        displayList(myList)
    elif choice == '2':
        traverseList(myList)
    elif choice == '3':
        index = int(input("Enter index to insert: "))
        element = input("Enter element to insert: ")
        insertElement(myList, index, element)
    elif choice == '4':
        element = input("Enter element to append: ")
        appendElement(myList, element)
    elif choice == '5':
        element = input("Enter element to remove: ")
        removeElement(myList, element)
    elif choice == '6':
```

```
        index = int(input("Enter index to replace: "))
        newElement = input("Enter new element: ")
        replaceElement(myList, index, newElement)
    elif choice == '7':
        element = input("Enter element to search: ")
        searchElement(myList, element)
    elif choice == '8':
        list_size(myList)
    elif choice == '9':
        print("Exiting program...")
        break
    else:
        print("Invalid choice. Please enter a valid option.")
```

Output:

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 4

Enter element to append: 59

Element 59 appended to the list

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 4

Enter element to append: 85

Element 85 appended to the list

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 3

Enter index to insert: 1

Enter element to insert: 69

Element 69 inserted at index 1

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 1

List: ['59', '69', '85']

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 3

Enter index to insert: 3

Enter element to insert: 61

Element 61 inserted at index 3

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 2

List elements:

59

69

85

61

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 5

Enter element to remove: 85

Element 85 removed from the list

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 6

Enter index to replace: 0

Enter new element: 16

Element at index 0 replaced with 16

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 8

Size of the list: 3

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 1

List: ['16', '69', '61']

1. Display list

2. Traverse list

3. Insert element

4. Append element

5. Remove element

6. Replace element

7. Search element

8. Size of list

9. Exit

Enter your choice: 9

Exiting program...

Conclusion: Data structures python has been studied and implemented.