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:
linked list

```
a = []
n = int(input('Enter the number of element:'))
for i in range (0, n):
    element = int(input('Enter the elements '))
    a.append(element)
print (a)

print('******************** Inserting Element **************************')
b = int(input('Enter the position in which you want to add the element:'))
e = int(input('Enter the element:'))
a.insert(b,e)
print(a)

print('******************** Append Element **************************')
c = int(input('Enter the Element to be Appended:'))
a.append(c)
print(a)

print('******************** Remove Element **************************')
d = int(input('Enter the Element to be Removed from the list:'))
a.remove(d)
print(a)

print('******************** Replace Element **************************')
e = int(input('Enter the position of element from the list:'))
f = int(input('Enter the element to be Replaced '))
g = int(input("with"))
a.remove(f)
a.insert(e,g)
print(a)

print('******************** Search Element **************************')
h = int(input('Enter the Element to be Searched from the list:'))
print("Element found at index:", a.index(h))
print(a)

print('******************** Size of Linked List Element **************************')
print(len(a))
```

Output:

```
C:\Deepak\Python_Exp\Python_Exp\.venv\Scripts\python.exe C:\Deepak\Python_Exp\Python_Exp\exp7.py
Enter the number of element:4
Enter the elements 1
Enter the elements 2
Enter the elements 3
Enter the elements 4
[1, 2, 3, 4]
******************** Inserting Element **************************
Enter the position in which you want to add the element:3
Enter the element:57
[1, 2, 3, 57, 4]
******************** Append Element **************************
Enter the Element to be Appended:58
[1, 2, 3, 57, 4, 58]
******************** Remove Element **************************
Enter the Element to be Removed from the list:57
[1, 2, 3, 4, 58]
******************** Replace Element **************************
Enter the position of element from the list:3
Enter the element to be Replaced 4
with59
[1, 2, 3, 59, 58]
******************** Search Element **************************
Enter the Element to be Searched from the list:58
Element found at index: 4
[1, 2, 3, 59, 58]
******************** Size of Linked List Element **************************
5

Process finished with exit code 0
```

Code:

```
a=[]
n = int(input("Enter No. of Elements: "))
for i in range (0,n):
   element= int(input("Elements in the stack: "))
   a.append(element)
   print(a)
print("********** Inserting element in the Stack *****************")
g=int(input("Enter Element to be added: "))
a.append(g)
print(a)
print("*********** Deleting element in the Stack **************")

print("\nDeleting First Element: ")
a.pop()
print(a)
print("\nDeleting Second Element: ")
a.pop()
print(a)
print("\nDeleting Third Element: ")
a.pop()
print(a)
print("\nlength of the Stack: ",len(a))
len(a)
print("\n********** Check if Stack is empty or not *********")
if (len == 0):
   print("stack is empty\n")
else:
   print("Stack is Full\n")

print("************ Search for the element *****************")
s=int(input("enter element to be searched: "))
print("Element found at index:", a.index(s))
```

Output:

```
C:\Users\Student\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Deepak\Python_Exp\Python_Exp\exp7.b.py
Enter No. of Elements: 5
Elements in the stack: 1
[1]
Elements in the stack: 2
[1, 2]
Elements in the stack: 3
[1, 2, 3]
Elements in the stack: 4
[1, 2, 3, 4]
Elements in the stack: 5
[1, 2, 3, 4, 5]
********** Inserting element in the Stack *****************
Enter Element to be added: 57
[1, 2, 3, 4, 5, 57]
*********** Deleting element in the Stack **************

Deleting First Element:
[1, 2, 3, 4, 5]

Deleting Second Element:
[1, 2, 3, 4]

Deleting Third Element:
[1, 2, 3]

length of the Stack:  3

********** Check if Stack is empty or not *********
Stack is Full
```

```
************* Search for the element *******************
enter element to be searched: 1
Element found at index: 0

Process finished with exit code 0
```

Conclusion: Hence, we successfully implemented data structures using built-in functions for linked lists, stacks, and queues in Python.