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:

```
a = []
n = int(input("Enter number of elements : "))
for i in range(0, n):
    element = int(input())
    a.append(element)
print(a)
print("*****************************")
print("Inserting element")
p = int(input("ENTER ELEMENT TO BE INSERTED: "))
f = int(input("ENTER POSITION: "))
a.insert(f,p)
print(a)
print("*****************************")
print("Removing element from the list")
r=int(input("Enter element to be Removed"))
a.remove(r)
print(a)
print("*****************************")
print("Replace element from the list")
k=int(input("Enter element position"))
h=int(input("Enter Element to be Repaced"))
g=int(input("with"))
a.remove(h)
```

```
a.insert(k,g)
print(a)
print("****************************")
print("Searching element")
u=int(input("Enter Element to be Searched"))
a.index(u)
print(u)
print("****************************")
print("Size of the LINKED lIST")
print(len(a))
```

OUTPUT:

```
C:\logs\pythonProject3\.venv\Scripts\python.exe C:\kashif\exp7.py
Enter number of elements : 5
1
2
3
4
5
[1, 2, 3, 4, 5]
****************************
Inserting element
ENTER ELEMENT TO BE INSERTED: 1
ENTER POSITION: 1
[1, 1, 2, 3, 4, 5]
****************************
Removing element from the list
Enter element to be Removed1
[1, 2, 3, 4, 5]
****************************
Replace element from the list
Enter element position2
Enter Element to be Repaced3
with2
[1, 2, 2, 4, 5]
****************************
Searching element
Enter Element to be Searched2
2
Size of the LINKED lIST
5
```

CODE:

```
print("Implementing Stack")
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("************************************")
print("Inserting element in the Stack")
g=int(input("Enter Element to be added"))
a.append(g)
print(a)
print("************************************")
print("Deleting element in the Stack")
print("Deleting First Element")
a.pop()
print(a)
print("Deleting Second Element")
a.pop()
print(a)
print("************************************")
print("length of the Stack",len(a))
len(a)
print("************************************")
print("Check if Stack is empty or not")
if (len == 0):
    print("stack is empty")
else:
```

```
    print("Stack is not Empty")
print("**********************************")
print("Search for the element")
s=int(input("enter element to be searched"))
k= a.index(s)
print(k)
```

OUTPUT:

```
Implementing Stack
Enter No. of Elements: 5
Elements in the stack: 2
[2]
Elements in the stack: 3
[2, 3]
Elements in the stack: 5
[2, 3, 5]
Elements in the stack: 69
[2, 3, 5, 69]
Elements in the stack: 5
[2, 3, 5, 69, 5]
**********************************
Inserting element in the Stack
Enter Element to be added5
[2, 3, 5, 69, 5, 5]
**********************************
Deleting element in the Stack
Deleting First Element
[2, 3, 5, 69, 5]
Deleting Second Element
[2, 3, 5, 69]
**********************************
length of the Stack 4
**********************************
Check if Stack is empty or not
Stack is not Empty
**********************************
Search for the element
enter element to be searched5
2
```