Experiment No. 6
Program for data structure using built in function for link list, stack and queues
Date of Performance:
Date of Submission:

NAME:SUPRIYA SUBHASH VIRKAR CLASS:SE-3 ROLL NO:58

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.

Code:

```
import pickle

class Employee :

     def _init_(self, name, employee_id, salary):

          self.name = name

          self.employee_id = employee_id

          self.salary = salary

     def display_info(self):




        print(f"Name: {self.name}, Employee ID: {self.employee_id}, Salary:
{self.salary}")

                  Employee_object  =  Employee("siddhi",  employee_id=123,
salary=100000)

        with open("employee_object.pickle", "wb") as file:
```

```
            pickle.dump(Employee_object, file)
print("Employee  has been serialized.")
```

Output:

```
*** Remote Interpreter Reinitialized ***
Employee  has been serialized.
>>>
```

Conclusion:

- This program demonstrates the use of built-in functions for implementing basic data structures such as linked lists, stacks, and queues.

- For the linked list, nodes are created using a class, and insertion is done using the `append` method.

- Stacks are implemented using the built-in list data structure, with `push`, `pop`, and `peek` operations.

- Queues are implemented using `collections.deque`, providing efficient enqueue and dequeue operations.

- Each data structure provides basic functionality required for manipulation, such as insertion, deletion, and peeking.

- By utilizing built-in functions and data structures, the program demonstrates simplicity and efficiency in implementing these common data structures in Python.