Computer Programming Laboratory

B.Tech. 1st Semester

Name : **SUBHENDU MAJI**

Roll Number : **18ETCS002121**

Department : **Computer Science and Engineering**

Faculty of Engineering & Technology

Ramaiah University of Applied Sciences

Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	1st Year / 1st Semester
Name of the Laboratory	Computer Programming Laboratory
Laboratory Code	18ESL109A

List of Experiments

1. Introduction to Python programming environment
2. Variables, data types, operators and expressions
3. Input output operations
4. Logic operations and decision making
5. Loop statements
6. Character and string operations
7. Functions
8. File handling
9. Data structures
10. Libraries

Name: SUBHENDU MAJI Roll Number: 18ETCS002121

Index Sheet

No.	Lab Experiment	Performing the experiment (7)	Document (7)	Viva (6)	Total Marks (20)
1	Introduction to Python programming environment				
2	Variables, data types, operators and expressions				
3	Input output operations				
4	Logic operations and decision making				
5	Loop statements				
6	Character and string operations				
7	Functions				
8	File handling				
9	Data structures				
10	Libraries				
11	Lab Internal Test conducted along the lines of SEE and valued for 50 Marks and reduced for 20 Marks				
	Total Marks				

Lab Internal Marks =

Signature of the Staff In-charge

Name: SUBHENDU MAJI Roll Number: 18ETCS002121

Laboratory 9

Title of the Laboratory Exercise: data structures

1. Introduction and Purpose of Experiment

   Data structure is a way of collecting and organizing data in such a way that various operations can be done on these data in an effective way. By solving these problems, students will become familiar with the implementations of stacks and queues.

2. Aim and Objectives

   Aim

   - To develop programs based on data structures

   Objectives

   At the end of this lab, the student will be able to

   - Use of appropriate data structure to store data
   - Create Python programs of basic data structures such as stacks and queues

3. Experimental Procedure

   i. Analyse the problem statement
   ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
   iii. Implement the algorithm in Python language
   iv. Execute the Python program
   v. Test the implemented program
   vi. Document the Results
   vii. Analyse and discuss the outcomes of the experiment

4. Questions

   i. Write a Python program to implement Queue operations
   ii. Write a Python program to implement Stack operations

5. Calculations/Computations/Algorithms

   5.1 algorithm of python program to implement stack operations

   Step1: start

Step2:create a class named stack

Step3:make a function push with command

self.item.append(x)

Step4: make a function pop with command

return self.item.pop()

step5:input element from user

step6:s.push(n)

step7:print s.item

step8:print s.pop() , s.item

step9: stop

5.2 algorithm of python program to implement Queue operations

Step1: start

Step2:create a class named queue

Step3:make a function enqueue with command

self.item.insert(0,x)

Step4: make a function dequeue with command

return self.item.pop()

step5:input element from user

step6:s.enqueue(n)

step7:print s.item

step8:print s.dequeue() , s.item

step9: stop

6. Presentation of Results

```
class stack:
    def __init__(self):
        self.item =[]
    def push(self,x):
        self.item.append(x)
    def pop(self):
        return self.item.pop()
    def empty(self):
        return self.item==[]
s=stack()
print('\npushing element in stack')
for iloop in range(0,6):
    n=input("enter element {} :".format(iloop+1))
    s.push(n)
    print('stack :',s.item)
print('\npoping element from stack')
for iloop in range(0,6):
    print('poping ',s.pop())
    print('stack : ',s.item)
```

Messages | Python Shell | Debug I/O

Debug I/O (stdin, stdout, stderr) appears below

```
pushing element in stack
enter element 1 :10
stack : ['10']
enter element 2 :9
stack : ['10', '9']
enter element 3 :8
stack : ['10', '9', '8']
enter element 4 :7
stack : ['10', '9', '8', '7']
enter element 5 :6
stack : ['10', '9', '8', '7', '6']
enter element 6 :5
stack : ['10', '9', '8', '7', '6', '5']

poping element from stack
poping  5
stack :  ['10', '9', '8', '7', '6']
poping  6
stack :  ['10', '9', '8', '7']
poping  7
stack :  ['10', '9', '8']
poping  8
stack :  ['10', '9']
poping  9
stack :  ['10']
poping  10
stack :  []
```

Figure 1 Python program to implement Queue operations

```python
class queue:
    def __init__(self):
        self.item =[]
    def enqueue(self,x):
        self.item.insert(0,x )
    def dequeue(self):
        return self.item.pop()
    def empty(self):
        return self.item==[]
s=queue()
print('\nenqueue element in queue')
for iloop in range(0,6):
    n=input("enter element {} :".format(iloop+1))
    s.enqueue(n)
    print('queue :',s.item)

print('\ndequeue element from queue')
for iloop in range(0,6):
    print('dequeue',s.dequeue())
    print('queue :',s.item)
```

Messages | Python Shell | Debug I/O

Debug I/O (stdin, stdout, stderr) appears below

```
enqueue element in queue
enter element 1 :9
queue : ['9']
enter element 2 :8
queue : ['8', '9']
enter element 3 :7
queue : ['7', '8', '9']
enter element 4 :6
queue : ['6', '7', '8', '9']
enter element 5 :5
queue : ['5', '6', '7', '8', '9']
enter element 6 :4
queue : ['4', '5', '6', '7', '8', '9']

dequeue element from queue
dequeue 9
queue : ['4', '5', '6', '7', '8']
dequeue 8
queue : ['4', '5', '6', '7']
dequeue 7
queue : ['4', '5', '6']
dequeue 6
queue : ['4', '5']
dequeue 5
queue : ['4']
dequeue 4
queue : []
```

Figure 2 Python program to implement Stack operations

7. Analysis and Discussions

   Stack and Queue ADTs are data structures in which there is restriction on the insertion and deletion of elements. the stack works in LIFO (last-in first-out) manner and queue works in FIFO (first-in first-out) manner.

8. Conclusions

   In a Queue, elements can be inserted at one end and can be removed only from the other end.

   In a Stack, elements can be inserted and removed only at one end.

9. Comments

   i. Limitations of Results

      The program can be made menu-driven. The illustration of stacks would have been better if the updated stack is printed vertically.