LAB MANUAL

DATA STRUCTURES LABORARTORY

Academic Year	:	2019 -2020
Course Code	:	ACSB05
Regulations	:	IARE –R18
Class	:	B. Tech IV Semester
Branch	:	AE EEE

Prepared by Mrs. A Jayanthi, Assistant Professor



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous) Dundigal, Hyderabad - 500 043



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

Vision

To bring forth professionally competent and socially sensitive engineers, capable of working across cultures meeting the global standards ethically.

Mission

To provide students with an extensive and exceptional education that prepares them to excel in their profession, guided by dynamic intellectual community and be able to face the technically complex world with creative leadership qualities.

Further, be instrumental in emanating new knowledge through innovative research that emboldens entrepreneurship and economic development for the benefit of wide spread community.

Quality Policy

Our policy is to nurture and build diligent and dedicated community of engineers providing a professional and unprejudiced environment, thus justifying the purpose of teaching and satisfying the stake holders.

A team of well qualified and experienced professionals ensure quality education with its practical application in all areas of the Institute.

Philosophy

The essence of learning lies in pursuing the truth that liberates one from the darkness of ignorance and Institute of Aeronautical Engineering firmly believes that education is for liberation.

Contained therein is the notion that engineering education includes all fields of science that plays a pivotal role in the development of world-wide community contributing to the progress of civilization. This institute, adhering to the above understanding, is committed to the development of science and technology in congruence with the natural environs. It lays great emphasis on intensive research and education that blends professional skills and high moral standards with a sense of individuality and humanity. We thus promote ties with local communities and encourage transnational interactions in order to be socially accountable. This accelerates the process of transfiguring the students into complete human beings making the learning process relevant to life, instilling in them a sense of courtesy and responsibility.



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad – 500043

	Program Outcomes
PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering
	fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex
	engineering problems reaching substantiated conclusions using first principles of mathematics, natural
	sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design
	system components or processes that meet the specified needs with appropriate consideration for
	the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research
	methods including design of experiments, analysis and interpretation of data, and synthesis of the
	information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern
	engineering and IT tools including prediction and modeling to complex engineering activities with
	an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal,
	health, safety, legal and cultural issues and the consequent responsibilities relevant to the
D 0 F	professional engineering practice.
PO/	Environment and sustainability : Understand the impact of the professional engineering solutions in
	societal and environmental contexts, and demonstrate the knowledge of, and need for
DOO	sustainable development.
PO8	Etnics : Apply etnical principles and commit to professional etnics and responsibilities and norms
DOO	of the engineering practice.
P09	diverse teams, and in multidicainlinemy settings
DO10	Communication: Communicate effectively on complex engineering estivities with the engineering
1010	communication. Communicate effectively on complex engineering activities with the engineering
	offective reports and design decumentation make affective presentations, and give and receive clear
	instructions
DO11	Instructions. Device transport and finance: Demonstrate knowledge and understanding of the angineering
FUIT	and management principles and apply these to one's own work as a member and leader in a team to
	and management principles and apply mese to one slowin work, as a memori and reduct in a team, to manage projects and in multidisciplinary environments
PO12	Life-long learning : Recognize the need for and have the preparation and ability to engage in
1012	independent and life-long learning in the broadest context of technological change
	independent and inchoing learning in the broadest context of technological challge.

Program Specific Autoomes - Aeronautical Engineering
1 rogram Specific Outcomes - Acronauticar Engineering
Professional skills: Able to utilize the knowledge of aeronautical/aerospace engineering in
innovative, dynamic and challenging environment for design and development of new products
Problem solving skills: imparted through simulation language skills and general purpose CAE
packages to solve practical, design and analysis problems of components to complete the challenge
of airworthiness for flight vehicles
Practical implementation and testing skills: Providing different types of in house and training and
industry practice to fabricate and test and develop the products with more innovative technologies.
Successful career and entrepreneurship: To prepare the students with broad aerospace knowledge
to design and develop systems and subsystems of aerospace and allied systems and become
technocrats
Program Specific Outcomes –Electrical and Electronics Engineering
Problem Solving: Exploit the knowledge of high voltage engineering in collaboration with power
systems in innovative, dynamic and challenging environment, for the research based team work.
Professional Skills: Identify the scientific theories, ideas, methodologies and the new cutting edge
technologies in renewable energy engineering, and use this erudition in their professional
development and gain sufficient competence to solve the current and future energy problems
universally.
Modern Tools in Electrical Engineering: Comprehend the technologies like PLC, PMC, process
controllers, transducers and HMI and design, install, test, maintain power systems and industrial
applications.



INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous) Dundigal, Hyderabad – 500043

ATTAINN	MENT OF PROGRA	AM OUTCOMES	& PROGRAM SPE	CIFIC OUTCOMES		
	AERO		EEE			
EXPT.No.	Program Outcomes Attained	Program Specific Outcomes Attained	Program Outcomes Attained	Program Specific Outcomes Attained		
I	PO1, PO2	PSO2	PO1, PO2	PSO1		
п	PO1, PO2, PO3	PSO2, PSO3	PO1, PO2, PO3	PSO1, PSO3		
ш	PO1, PO2, PO3	PSO2, PSO3	PO1, PO2, PO3	PSO1, PSO3		
IV	PO1, PO2, PO3	PSO2, PSO3	PO1, PO2, PO3	PSO1, PSO3		
V	PO1, PO2, PO3	PSO2, PSO3	PO1, PO2, PO3	PSO1, PSO3		
VI	PO1, PO2, PO3	PSO2, PSO3	PO1, PO2, PO3	PSO1, PSO3		
VII	PO1, PO2, PO3, PO5	PSO2, PSO3	PO1, PO2, PO3, PO5	PSO1, PSO3		
VIII	PO1, PO2, PO3	PSO2, PSO3	PO1, PO2, PO3	PSO1, PSO3		
IX	PO1, PO2, PO3, PO5	PSO2, PSO3	PO1, PO2, PO3, PO5	PSO1, PSO3		
X	PO1, PO2, PO3, PO5	PSO2 ,PSO3	PO1, PO2, PO3, PO5	PSO1, PSO3		
XI	PO1, PO2, PO3, PO5	PSO2, PSO3	PO1, PO2, PO3, PO5	PSO1, PSO3		
XII	PO1, PO2, PO3, PO5	PSO2, PSO3	PO1, PO2, PO3, PO5	PSO1, PSO3		

(Autonomous) Dundigal, Hyderabad - 500 043					
	Certificate				
This is to Certify that it i	s a bonafied record of Practical work done by				
Sri/Kum	bearingthe				
Roll No	of Clas				
	Branch in th				
	laboratory during the Academic				
year	under our supervision.				
Head of the Department	Lecture In-Charge				
External Examiner	Internal Examiner				



S. No.	List of Experiments	Page No.	Date	Remarks
Ι	BASICS OF PYTHON			
II	SEARCHING TECHNIQUES			
III	SORTING TECHNIQUES			
IV	IMPLEMENTATION OF STACK AND QUEUE			
V	APPLICATIONS OF STACK			
VI	IMPLEMENTATION OF SINGLE LINKED LIST			
VII	IMPLEMENTATION OF CIRCULAR SINGLE LINKED LIST			
VIII	IMPLEMENTATION OF DOUBLE LINKED LIST			
IX	IMPLEMENTATION OF STACK USING LINKED LIST			
Х	IMPLEMENTATION OF QUEUE USING LINKED LIST			
XI	GRAPH TRAVERSAL TECHNIQUES			
XII	IMPLEMENTATION OF BINARY SEARCH TREE			

DATA STRUCTURES LABORATORY

Cour	se Code	Category	Ho	urs / \	Week	Credits	Ma	aximum N	Aarks
Δ(~SB05	Coro	L	Т	Р	С	CIA	SEE	Tota
		Core	-	-	3	1.5	30	70	100
Contact	Classes: Nil	Tutorial Classes: Nil	P	ractic	al Class	ses: 36	То	tal Classe	es: 36
COURSE The course	OBJECTIVE should enable	S: the students to:							
I. Unders II. Implem III. Analyz IV. Develo V. Identify	tand various d nent linear and e various algor p real-time app y suitable data	ata representation techniq non-linear data structures rithms based on their time plications using suitable d structure to solve various	ues in s. e and s lata str comp	the re pace of ructure outing	al worl complex c. problen	d. kity. ns.			
		LIST OF	EXPI	ERIM	ENTS				
Week -1	BASICS OF	PYTHON							
Write Pytho a. To find b. To print c. To find	on programs fo the biggest of the Fibonacci GCD of two n	r the following: given n numbers using co series using functions umbers	ontrol s	statem	ents and	dlists			
Week -2	SEARCHIN	NG TECHNIQUES							
Write Pyth in ascendin a. Linear s b. Binary s	on programs fo g order. earch search	or implementing the follo	wing s	searchi	ng tech	niques to ar	range a I	list of inte	gers
Week -3	SORTING 7	TECHNIQUES							
Write Pyth in ascendin a. Bubble b. Insertion c. Selectio	on programs fo g order. sort n sort n sort	or implementing the follo	wing s	sorting	technic	ques to arrar	nge a list	ofinteger	S
Week -4	IMPLEME	NTATION OF STACK	AND	QUE	JE				
Write Pyth a. Design b. Design	on programs to and implement and implement	o for the following: t Stack and its operations t Queue and its operations	using s using	List. g List.					
Week -5	APPLICAT	TONS OF STACK							
Write Pyth	on programs for	or the following:	n into	postfi	x expre	ssion			

Week-6	IMPLEMENTATION OF SINGLE LINKED LIST
Write Pytho (i) Creation	n programs for the following operations on Single Linked List. (ii) insertion (iii) deletion (iv) traversal
Week -7	IMPLEMENTATION OF CIRCULAR SINGLE LINKED LIST
Write Pythe (i) Creation	on programs for the following operations on Circular Linked List. (ii) insertion (iii) deletion (iv) traversal
Week -8	IMPLEMENTATION OF DOUBLE LINKED LIST
Write Pythe (i) Creation	on programs for the following operations on Double Linked List. (ii) insertion (iii) deletion (iv) traversal in both ways.
Week -9	IMPLEMENTATION OF STACK USING LINKED LIST
Write a Pyt	hon program to implement Stack using linked list.
Week -10	IMPLEMENTATION OF QUEUE USING LINKED LIST
Write a Pyt	hon program to implement Linear Queue using linked list.
Week -11	GRAPH TRAVERSAL TECHNIQUES
Write Pytho a. Depth fi b. Breadth	on programs to implement the following graph traversal algorithms: rst search. first search.
Week -12	IMPLEMENTATION OF BINARY SEARCH TREE
Write a Pyt	hon program to perform the following:
a. Create a	binary search tree.
c. Count th	he number of nodes in the binary search tree.
LIST OF I	REFERENCE BOOKS:
1. Rance 2011	D.Necaise, "Data Structures and Algorithms using Python", Wiley, John Wiley&Sons, INC.,
2. Benjan	in Baka, David Julian, "Python Data Structures and Algorithms", Packt Publishing Ltd., 2017.
WEB REF	ERENCES:
1. https://d	ocs.python.org/3/tutorial/datastructures.html
2. http://in	eractivepython.org/runestone/static/pythonds/index.html
3. http://ww	ww.tutoriaispoint.com/data_structures_algorithms
5. http://ww	ww.studytonight.com/data-structures/
6. http://w	ww.coursera.org/specializations/data-structures-algorithms
7. http://cs	e01-iiith.vlabs.ac.in/

WEEK – 1

BASICS OF PYTHON

OBJECTIVE:

- a. Write a Python script to find the biggest of the given numbers using control statements and lists
- b. Write a Python script to print the Fibonacci series using functions.
- c. Write a Python script to find the GCD of two numbers.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Biggest of the given numbers using control statements and list:

- 1. Read a list of integers.
- 2. Assume the first number as maximum number.
- 3. Compare each number n with the maximum number and if n is bigger than max then change max with n.
- 4. Repeat this process for all numbers.
- 5. Return max

Fibonacci series using function:

- 1. Read number of terms n.
- 2. Send n to recursive method recur_fibo()
- 3. if $n \le 1$ then return n
- 4. otherwise return(recur_fibo(n-1) +recur_fibo(n-2))

GCD of two numbers:

- 1. Read two integers n1 andn2.
- 2. Send n to recursive method compute GCD(n1,n2).
- 3. Find the smaller number by checking if n1 > n2 then smaller = n2, otherwise smaller =n1
- 4. for each number i, compute if ((n1 % i == 0) and (n2 % i == 0)) then gcd =i
- 5. return gcd

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .py extension.
- b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

Biggest of the given numbers using control statements and list:

Driver code arr=[3,2,4,1,5,8,6,9,7] large(arr)

Output:

- 0 ×

-	
🛃 Python 3.7.3 Shell	
File Edit Shell Debug Options Window Help	
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win3. Type "help", "copyright", "credits" or "license()" for more information.	2

RESTART: C:\Users\Sirisha\AppData\Local\Programs\Python\Python37-32\Scripts\largest_list.py largest element is 9 >>>



GCD of two numbers:

def computeGCD(x,y):

```
# choose the smaller number
if x > y:
    smaller = y
else:
    smaller = x
for i in range(1, smaller+1):
    if((x % i == 0) and (y % i == 0)):
        gcd=i
```

return gcd

#DriverCode

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

print("The GCD of", num1,"and", num2,"is", computeGCD(num1, num2))

Output:



PRE LAB VIVA QUESTIONS:

- a. What is Python? What are the benefits of usingPython?
- b. How memory is managed inPython?
- c. In Python what isslicing?
- d. What are the different ways of accessing elements in alist?
- e. State any five built-in functions used inlists?

LAB ASSIGNMENT:

- a. Write a Python program to find the factors of anumber?
- b. Write a Python program to find the factorial of a number using recursion?

- c. Write a Python program to check if the input number is prime or not?
- d. Write a Python program to find the sum of natural numbers up to n using recursive function?
- e. Write a Python program to display all the prime numbers within an interval?

POST LAB VIVA QUESTIONS:

- a. What is the difference between list and tuple?
- b. What are the built-in type does python provides?
- c. State the built-in set operators?
- d. Define class, object, attribute and method?
- e. What is lambda in Python?

WEEK – 2

SEARCHING TECHNIQUES

OBJECTIVES:

a. Write a Python script to implement linear search technique.

b. Write a Python script to implement binary search technique.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Linear search technique:

- Given a list of n elements and search a given element x in the list using linear search.
 - a. Start from the leftmost element of list a[] and one by one compare x with each element of lista[].
 - b. If x matches with an element, return theindex.
 - a. If x doesn't match with any of elements, return-1.

Binary search technique:

Given a sorted list of a[] of n elements, search a given element x in list.

- a. Search a sorted list by repeatedly dividing the search interval in half. Begin withan interval covering the wholelist.
- b. If the search key is less than the item in the middle item, then narrowthe interval to the lower half. Otherwise narrow it to the upperhalf.
- c. Repeat the procedure until the value is found or the interval isempty.

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.

b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

Linear search technique:

```
def linear_search(obj, item):
    for i in range(len(obj)):
        if obj[i] == item:
            return i
```

return -1

#Driver code
arr=[1,2,3,4,5,6,7,8]
x=int(input("what are you searching for?"))
result=linear_search(arr,x)

if result==-1: print ("element does not exist")

else:

print ("element exist in position %d" %result)

Output:

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32 Type "help", "copyright", "credits" or "license()" for more information.

RESTART: C:\Users\Sirisha\AppData\Local\Programs\Python\Python37-32\Scripts\linear_search.py what are you searching for?5 element exist in position 4



R^R ∧ 🕞 ⊄× 🦟 ENG 10 اسا 2019

o ×

```
Binary search technique:
array =[1,2,3,4,5,6,7,8,9]
```

```
def binary_search(searchfor,array):
    lowerbound=0
    upperbound=len(array)-1
    found=False
    while found==False and lowerbound<=upperbound:
        midpoint=(lowerbound+upperbound)//2
        if array[midpoint]==searchfor:
            found=True
            return found
        elif array[midpoint]<searchfor:
            lowerbound=midpoint+1
        else:
            upperbound=midpoint-1
        return found</pre>
```

```
#Driver code
searchfor=int(input("what are you searching for?"))
if binary_search(searchfor,array):
    print ("element found")
else:
    print ("element not found")
```

Output:



PRE LAB VIVA QUESTIONS:

- a. Define searching process?
- b. How many types of searching are there?
- c. Why binary search method is more efficient then liner search?
- d. What is worse case?

LAB ASSIGNMENT:

- a. A person has registered for voter id, he received a voter number and he need to check whether it exist in the voter or not. Use a binary searching in a recursive way to find whether the voter number exist in the list ornot.
- b. Use linear search technique to search for a key value in a given list of characters and print the message found or not.

POST LAB VIVAQUESTIONS:

- a. What do you understand by the term "linear search is unsuccessful"?
- b. Efficiency of linear search?
- c. What is the drawback of linear search?

WEEK – 3 SORTING TECHNIQUES

OBJECTIVES:

- a. Write a Python script to implement bubble sort.
- b. Write a Python script to implement insertion sort.
- c. Write a Python script to implement selection sort.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Bubble sort:

- 1. Starting with the first element(index = 0), compare the current element with the next element of the array.
- 2. If the current element is greater than the next element of the array, swap them.
- 3. If the current element is less than the next element, move to the next element. Repeat Step1.

Insertion sort:

- 1. It is efficient for smaller data sets, but very inefficient for larger lists.
- 2. Insertion Sort is adaptive, that means it reduces its total number of steps if a partially sorted array is provided as input, making it efficient.
- 3. It is better than Selection Sort and Bubble Sort algorithms.
- 4. Its space complexity is less. Like bubble Sort, insertion sort also requires a single additional memory space.
- 5. It is a stable sorting technique, as it does not change the relative order of elements which are equal.

Selection sort:

- 1. Starting from the first element, we search the smallest element in the array, and replace it with the element in the first position.
- 2. We then move on to the second position, and look for smallest element present in the subarray, starting from index 1, till the last index.
- 3. We replace the element at the second position in the original array, or we can say at the first position in the subarray, with the second smallest element.
- 4. This is repeated, until the array is completely sorted.

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .py extension.
- b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

Bubble sort:

def bubbleSort(arr): n = len(arr)

Traverse through all array elements
for i in range(n):

Last i elements are already in place for j in range(0, n-i-1):

traverse the array from 0 to n-i-1
Swap if the element found is greater
than the next element
if arr[j] > arr[j+1] :
 arr[j], arr[j+1] = arr[j+1], arr[j]

```
# Driver code to test above
arr = [64, 34, 25, 12, 22, 11, 90]
```

bubbleSort(arr)

print ("Sorted array is:") for i inrange(len(arr)): print ("%d"%arr[i])

Output:



Insertion sort: def insertionSort(arr):

> # Traverse through 1 to len(arr) for i in range(1, len(arr)): key = arr[i]

Move elements of arr[0..i-1], that are # greater than key, to one position ahead # of their currentposition j = i-1 while j >=0 and key < arr[j] : arr[j+1] = arr[j]j -= 1 arr[j+1] = key

Driver code to test above arr = [12, 11, 13, 5, 6] insertionSort(arr) print ("Sorted array is:") for i in range(len(arr)): print ("%d" %arr[i])

Output:





PRE LAB VIVA QUESTIONS:

- a. Explain the term sorting?
- b. What are the different types of sorts in data structures?
- c. Define the bubble sort?
- d. Define the insertion sort?
- e. Define the selection sort?

LAB ASSIGNMENT:

- a. Formulate a program that implement Bubble sort, to sort a given list of integers in descending order.
- b. Compose a program that implement Insertion sort, to sort a given list of integers in descending order.
- c. Write a program that implement Selection sort, to sort a given list of integers in ascending order.
- d. Formulate a program to sort N names using selection sort.
- e. Write a program to sort N employee records based on their salary using insertion sort.
- f. A class contains 50 students who acquired marks in 10 subjects write a program to display top10 students roll numbers and marks in sorted order by using bubble sorting technique.

POST LAB VIVA QUESTIONS:

- a. How many passes are required in selection sort?
- b. Write the time complexity of insertion sort?
- c. Write the time complexity of selection sort?
- d. Write the time complexity of bubble sort?

WEEK - 4

IMPLEMENTATION OF STACKS AND QUEUES

OBJECTIVES:

a. Write a Python script to design and implement stack and its operations using list.

b. Write a Python script to design and implement queue and its operations using list.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Stack and its operations using list:

- a. Stack is a linear data structure which works under the principle of last in first out. Basic operations: push, pop, display.
- b. PUSH: if (top==MAX), display Stack overflow. Otherwise reading the data and making stack[top] =data and incrementing the top value by doing top++.
- c. Pop: if (top==0), display Stack underflow. Otherwise printing the element at the top of the stack and decrementing the top value by doing that op.
- d. DISPLAY: If (top==0), display Stack is empty. Otherwise printing the elements in the stack from stack [0] to stack [top].

Queue and its operations using list:

- a. Queue is a linear data structure which works under the principle of first in first out. Basic operations: Insertion, deletion, display.
- b. Insertion: if (rear==MAX), display Queue is full. Else reading data and inserting at queue [rear], and doing rear++.
- c. Deletion: if (front==rear), display Queue is empty. Else printing element at queue [front] and doing front++.
- d. Display: if (front==rear) ,display No elements in the queue .Else printing the elements from queue[front] to queue[rear].

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .py extension.
- b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

Stack and its operations using list:

Function to create a stack. It initializes size of stack as 0

def createStack():
 stack = []
 return stack

Stack is empty when stack size is 0 def isEmpty(stack):

```
return len(stack) == 0
```

Function to add an item to stack. It increases size by 1 def push(stack, item):

```
if(len(stack)==size):
print("overflow")
return
stack.append(item)
```

```
# Function to remove an item from stack. It decreases size by 1
def pop(stack):
    if (isEmpty(stack)):
        print("underflow")
```

```
return
         return stack.pop()
     #Function to know peek element
     def peek(stack):
            if(isEmpty(stack)):
                  print("stack empty")
                  return
            else:
                  n=len(stack)
                  print("peek element is: ",stack[n-1])
     #Function to display stack
     def display(stack):
            print(stack)
     # Driver program to test above functions
     stack = createStack()
     size=int(input("enter the size of stack"))
     print("Menu\n1.push(p)\n2.pop(o)\n3.peek(e)")
     choice=1
     while choice!='q':
            print("enter your choice")
            ch=input()
            choice=ch.lower()
            if choice=='p':
                  push(stack,int(input("enter a value")))
                  display(stack)
            elif choice=='o':
                  pop(stack)
                  display(stack)
            elif choice=='e':
                 peek(stack)
            else:
                  print("enter proper choice or q - quit")
Output:
Prython 3.7.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3-refleec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
 >>>>
RESTART: C/Users/Sirisha/AppData/Local/Programs/Python/Python37-32/Scripts/stacl_list.py
enter the size of stack5
Memu
Lpush(p)
200000
 2.pop(o)
3.peek(e)
enter your choice
p
enter a value1
[1]
enter your choice
enter your choice

p

enter a value2

[1, 2]

enter your choice

p

enter a value3

[1, 2, 3]

enter your choice

p

enter a value4

[1, 2, 3, 4]

enter your choice

p
 enter a value5
[1, 2, 3, 4, 5]
enter your cho
p
enter a value6
 overflow
[1, 2, 3, 4, 5]
                   ၔ 🗈 🗖 🦉
                                                                                                                           x<sup>R</sup> ∧ ≌ ⊄× ∉ ENG 11:56 📮
 日 오 日
```

-Ø ×

20 | Page

Python 3.7.3 Shell	- 0 ×
Eile Edit Shell Debug Options Window Help	
enter your choice	·
p	
enter a value4	
[1, 2, 3, 4]	
enter your choice	
p	
enter a values	
[1, 2, 3, 4, 5]	
anter your choice	
penter a value6	
overflow	
enter your choice	
0	
[1, 2, 3, 4]	
enter your choice	
0	
[1, 2, 3]	
enter your choice	
0	
[1, 2]	
enter your choice	
3	
Li enter vour choice	
underflow	
Π	
enter your choice	
	Ln: 55 Col:
	11:58

Queue and its operations using list:

```
def enqueue(a,item):
   globalr
   global f
   if r==-1 and f==-1:
       r=0
       f=0
       a.insert(r,item)
   elif r == (n-1):
       print("overflow")
       return
   else:
       r+=1
       a.insert(r,item)
   display(a)
def dequeue(a):
   global r
   global f
   if r==(n-1) and f==(n-1):
       item=a[f]
       r=-1
       f=-1
   elif r==-1 and f==-1:
       print("underflow")
       return
   else:
       item=a[f]
       f+=1
   print("deleted item is:",item)
   display(a)
```

```
def display(a):
    print("\ncurrent queue is:")
    for i in range(f,r+1):
        if f==-1 and r==-1:
            print("Queue is empty!")
            return
        print(a[i],end=" ")
```

#DC

n=int(input("enter the size of list")) a=[] r=-1 f=-1 print("Menu\n1.enqueue(e)\n2.dequeue(d)\n3.exit(q)")

```
choice=1
while choice!='q':
    print("enter your choice")
    ch=input()
    choice=ch.lower()
    ifchoice=='e':
        enqueue(a,int(input("enter a value")))
        display(a)
elif choice=='d':
        dequeue(a)
        display(a)
```

print("enter proper choice")

else:

Output:



PRE LAB VIVA QUESTIONS:

- a. What is stack?
- b. What are the operations performed on stack?

- c. How stacks are implemented?
- d. What are the applications of stack?
- e. What is recursion?
- f. Define "Top of stack".
- g. How to implement stack?
- h. Define a queue?
- i. Define the condition "overflow".
- j. Define the condition "underflow".
- k. Define a queue.
- 1. Which principle is followed in queue?
- m. List out the applications of queue?

LAB ASSIGNMENT

- a. Write a program to implement stack and its operations using arrays.
- b. Formulate a program to reverse a list of numbers using stack.
- c. Write a program to find the factorial of a number using stack.
- d. Develop a program to check a given expression is balanced or not using stack
- e. Compose a program to implement Queue operations using arrays.
- f. Formulate a program to implement circular queue operations using arrays.
- g. Write a program to implement a priority queue?

POST LAB VIVA QUESTIONS:

- a. Write the time complexity of PUSH operation?
- b. Write the time complexity of POP operation?
- c. List out the applications of stack?
- d. How to remove an element from stack?
- e. How to insert an element into a stack?
- f. Write the time complexity to insert an element in to a queue?
- g. Write the time complexity to delete an element from a queue?
- h. List out the advantage of circular queue over linear queue?
- i. Define a priority queue?
- 1. Define DEQUE?

WEEK - 5

APPLICATIONS OF STACKS

OBJECTIVES:

- a. Write a Python script that uses stack operations to convert infix expression to postfix expression.
- b. Write a Python script that uses stack operations for evaluating the postfix expression.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Infix expression to postfix expression:

Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

- 1. Push "("onto Stack, and add ")" to the end of X.
- 2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
- 3. If an operand is encountered, add it to Y.
- 4. If a left parenthesis is encountered, push it onto Stack.
- 5. If an operator is encountered, then:
 - 1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.
 - 2. Add operator to Stack. [End off]
- 6. If a right parenthesis is encountered, then:
 - 1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.
 - 2. Remove the left Parenthesis. [End ofIf] [End ofIf]
- 7. END.

Evaluation of the postfix expression:

- 1. Create a stack to store operands (orvalues).
- 2. Scan the given expression and do following for every scanned element.
- 1. If the element is a number, push it into the stack
- 2. If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack
- 3. When the expression is ended, the number in the stack is the final answer

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .py extension.
- b. Execute : Go to Run ->Run module(F5)

SOURCE CODE: Infix expression to postfix expression:

import string class Conversion:

Constructor to initialize the class variables
def_init_(self, capacity):
 self.top = -1
 self.capacity = capacity
 # This array is used a stack
 self.array = []
 # Precedence setting
 self.output = []
 self.precedence = {'+':1, '-':1, '*':2, '/':2, '^':3}

```
# check if the stack is empty
def isEmpty(self):
  return True if self.top == -1 else False
# Return the value of the top of the stack
def peek(self):
  return self.array[-1]
# Pop the element from the stack
def pop(self):
  if not self.isEmpty():
     self.top -= 1
     return self.array.pop()
  else:
     return "$"
# Push the element to the stack
def push(self, op):
  self.top += 1
  self.array.append(op)
# A utility function to check is the given character
# is operand
def isOperand(self, ch):
  return ch.isalpha()
# Check if the precedence of operator is strictly
# less than top of stack or not
def notGreater(self, i):
  try:
     a = self.precedence[i]
     b = self.precedence[self.peek()]
     return True if a <= b else False
  except KeyError:
     return False
# The main function that converts given infix expression
# to postfix expression
def infixToPostfix(self, exp):
  # Iterate over the expression for conversion
  for i in exp:
     # If the character is an operand,
     # add it to output
     if self.isOperand(i):
        self.output.append(i)
     # If the character is an '(', push it to stack
     elif i == '(':
        self.push(i)
     # If the scanned character is an ')', pop and
     # output from the stack until and '(' is found
     elif i ==')':
        while( (not self.isEmpty()) and self.peek() != '('):
          a = self.pop()
          self.output.append(a)
        if (not self.isEmpty() and self.peek() != '('):
          return -1
```

```
else:
  self.pop()
```

```
# An operator is encountered
else:
  while(not self.isEmpty() and self.notGreater(i)):
     self.output.append(self.pop())
  self.push(i)
```

```
# pop all the operator from the stack
while not self.isEmpty():
  self.output.append(self.pop())
```

```
result= "".join(self.output)
    print(result)
# Driver program to test above function
exp = "a+b*(c^d-e)^{f+g*h}-i"
obj = Conversion(len(exp))
obj.infixToPostfix(exp)
```

Output:

Python 3.7.3 Shell
 File Edit Shell Debug Options Window Help
 Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
 Type "help", "copyright", "credits" or "license()" for more information.

RESTART: C:\Users\Sirisha\AppData\Local\Programs\Python\Python37-32\Scripts\infix_to_postfix.py abcd^e-fgh*+^*+i->>>

🖽 🔎 🖬 🤮 📄 🗔 🧔

x^R ∧ I⊡ ⊄× //; ENG 11:25

đ ×

Evaluation of the postfix expression: class Evaluate:

Constructor to initialize the class variables def init (self, capacity): self.top = -1self.capacity = capacity # This array is used a stack self.array = [] # check if the stack is empty

def isEmpty(self): return True if self.top == -1 else False

Return the value of the top of the stack

```
def peek(self):
    return self.array[-1]
# Pop the element from the stack
def pop(self):
    if not self.isEmpty():
        self.top -= 1
        return self.array.pop()
    else:
        return "$"
```

```
# Push the element to the stack
def push(self, op):
    self.top += 1
    self.array.append(op)
```

```
# The main function that converts given infix expression
# to postfix expression
def evaluatePostfix(self, exp):
```

```
# Iterate over the expression for conversion for i in exp:
```

```
# If the scanned character is an operand
# (number here) push it to the stack
if i.isdigit():
    self.push(i)
```

```
# If the scanned character is an operator,
# pop two elements from stack and apply it.
else:
    val1 = self.pop()
    val2 =self.pop()
    self.push(str(eval(val2 + i +val1)))
```

```
returnint(self.pop())
```

```
# Driver program to test above function
exp = "231*+9-"
obj = Evaluate(len(exp))
print ("Value of {0} is {1}".format(exp, obj.evaluatePostfix(exp)))
```

Output:

Python 3.7.3 Shell

Elle Edit Shell Debug Options Window Help Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32 Type "help", "copyright", "credits" or "license()" for more information.

RESTART: C:/Users/Sirisha/AppData/Local/Programs/Python/Python37-32/Scripts/Eval_postfix.py Value of 231*+9- is -4



📌 ^ 🗖 🕸 🌾 ENG 🕺 🕅 🕅 =

۵ ×

PRE-LAB VIVA QUESTIONS:

- a. What is an expression?
- b. Which operator is having highest priority?
- Give an example for prefix expression? c.
- d. Give an example for postfix expression?

LAB ASSIGNMENT:

- a. Formulate a program to convert infix expression into postfix expression.
- b. Write a program to evaluate any postfix expression.
- c. Compose a program to convert infix expression into prefix expression.
- d. Write a program to convert prefix expression into postfix expression.
- Write a program to evaluate any prefix expression. e.

POST-LAB VIVA QUESTIONS:

- What is the output of the following expression: 2345 + *a.
- What is the advantage of postfix expression? b.
- What is the maximum difference between number of operators and operands? c.
- Which expression doesn't require parenthesis? d.
- What is the output of the following expression: + * 2 3 45 e.

WEEK-6

IMPLEMENTATION OF SINGLE LINKED LIST

OBJECTIVES:

Write Python programs for the following operations on Single Linked List.

(i) Creation (ii) insertion (iii) deletion (iv) traversal

RESOURCE:

Python 3.7.3

PROGRAM LOGIC: Single Linked List: (i) creation (ii) insertion (iii) deletion (iv) traversal

- (i) Creation
- 1. first=new node;{createthe1stnode of the list pointed by first};
- 2. Read(Data(first));
- 3. NEXT(First)=NULL;
- 4. Far a First; [point Far to the First]
- 5. For I=1 to N-1 repeat steps 6 to 10
- 6. X=new node;
- 7. Read(Data(X))
- 8. NEXT(X)=NULL;
- 9. NEXT(Far)=X; {connect the nodes}
- 10. Far=X;[shift the pointer to the last node of the list]
- 11. [end of For Loop]
- 12. END

(ii) Insertion

Empty list case: When list is empty, which is indicated by (head == NULL) condition, the insertion is quite simple. Algorithm sets both head and tail to point to the new node.

Add first: In this case, new node is inserted right before the current head node.

It can be done in two steps:

- 1. Update the next link of a new node, to point to the current head node.
- 2. Update head link to point to the new node.

Add last: In this case, new node is inserted right after the current tail node.

It can be done in two steps:

- 1. Update the next link of the current tail node, to point to the new node
- 2. Update tail link to point to the new node.

General case: In general case, new node is always inserted between two nodes, which are already in the list. Head and tail links are not updated in this case.

Such an insert can be done in two steps:

- 1. Update link of the "previous" node, to point to the new node.
- 2. Update link of then new node, to point to the "next" node.

(iii) Deletion

List has only one node: When list has only one node, which is indicated by the condition, that the head points to the same node as the tail, the removal is quite simple. Algorithm disposes the node, pointed by head (or tail) and sets both head and tail to *NULL*.

Remove first: In this case, first node (current head node) is removed from the list.

It can be done in two steps:

- 1. Update head link to point to the node, next to the head.
- 2. Dispose removed node.

Remove last: In this case, last node (current tail node) is removed from the list. This operation is a bit trickier, than removing the first node, because algorithm should find a node, which is previous to the tail first.

It can be done in three steps:

- 1. Update tail link to point to the node, before the tail. In order to find it, list should be traversed first, beginning from thehead.
- 2. Set next link of the new tail toNULL.
- 3. Dispose removed node.

General case: In general case, node to be removed is always located between two list nodes. Head and tail links are not updated in this case.

Such a removal can be done in two steps:

- 1. Updatenextlinkofthepreviousnode,topointtothenextnode, relative totheremovednode.
- 2. Dispose removed node.

(iv) Traversal

- 1. If First=NULL then {print "List empty"STOP};
- 2. count=0;
- 3. ptr=First; {point ptr to the 1^{st} node}
- 4. While ptr<>NULL repeat Steps 5 to6
- 5. count=count+1;
- 6. ptr=NEXT(ptr) [shift ptr to the nextnode]
- 7. print ('Number of nodes=',count)
- 8. END

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.
- b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

class Node: def_init_(self,data): self.data=data self.next=None

class Sll:

def_init_(self):
 self.start=None
defcreatelist(self):
 n=int(input("enter number of node"))
 for i in range(n):

```
data=int(input("enter value"))
       newnode=Node(data)
       if self.start==None:
          self.start=newnode
       else:
          temp=self.start
          while temp.next!=None:
              temp=temp.next
          temp.next=newnode
def insertend(self):
   n=int(input("enter value"))
   newnode=Node(n)
   if self.start==None:
       self.start=newnode
   else:
       temp=self.start
       while temp.next!=None:
          temp=temp.next
       temp.next=newnode
def insertmid(self):
   n=int(input("enter value"))
   newnode=Node(n)
   pos=int(input("enter position"))
   c=self.count()
   if self.start==None:
       self.start=newnode
   else:
       if pos>1 and pos<=c:
          temp=self.start
          prev=temp
          i=1
           while i<pos:
              prev=temp
              temp=temp.next
              i=i+1
       prev.next=newnode
       newnode.next=temp
def count(self):
   nc=0
   temp=self.start
   while temp!=None:
       nc+=1
       temp=temp.next
   print("number of nodes=%d" %nc)
   return nc
def deletemid(self):
   count=1
   if self.start==None:
       print("empty")
   else:
       position=int(input("enter position"))
       c=self.count()
       if position>c:
          print("check position")
       if position>1 and position<c:
          temp=prev=self.start
           while count<position:
              rev=temp
```

```
temp=temp.next
                  count=count+1
              prev.next=temp.next
              del temp
           else:
              print("check position")
   def deleteend(self):
       global prev
       if self.start==None:
           print("empty")
       else:
           temp=self.start
           prev=self.start
           while temp.next!=None:
              prev=temp
              temp=temp.next
           prev.next=None
           del temp
   def insertbegin(self):
       n=int(input("enter value"))
       newnode=Node(n)
       if self.start==None:
           self.start=newnode
       else:
           temp=self.start
          newnode.next=temp
          self.start=newnode
   def deletebegin(self):
       global prev
       if self.start==None:
           print("empty")
       else:
          temp=self.start
          newstart=self.start.next
          del temp
          self.start=newstart
   def display(self):
       print("elements in single linked list are:")
       if self.start==None:
          print("empty")
       else:
           temp=self.start
           print("%d" %(temp.data))
           while temp.next!=None:
              temp=temp.next
              print("%d" %(temp.data))
### OUTSIDE CLASS
def menu():
```

print("1. create list n2. insert begin n3. insertend n4. insertmid n5. deletebegin n6. deleteend n7. deletemid n8. count n9. display n10. exit")

def stop():

print("u r about to terminate program")
exit()

s=Sll()

def default(): print("check ut input")

menu() whileTrue: menu={ 1: s.createlist, 2: s.insertbegin, 3:s.insertend, 4:s.insertmid, 5: s.deletebegin, 6:s.deleteend, 7:s.deletemid, 8: s.count, 9: s.display, 10: stop} option=int(input("enter ur choice")) menu.get(option)()

Output:

Python 3.7.3 Shell* File Edit Shell Debug Options Window Help Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32 Type "help", "copyright", "credits" or "license()" for more information. 22 Ø X Second 10. exit enter ur choice1 enter number of node3 enter value15 enter value25 enter value25 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice Ln: 25 Col: 15 يري A^R ∧ ⊫ ⊄× ⁄⁄_c ENG 11:17 11 Jul 2019 🔳 🔉 🗄 😂 📄 🧔 🕅 F

Python 37.3 (c).3.3 adfectedal 2. Mar 25 2019, 21:6533 (Mar C v 1916 32 fr ufficial) on wais2 Python 7.7 (C).4007 Sixinka AypDate Local Programs/Python 7-32 Script Simple_LL py 1. errore lat 2. mort Design 2. mort Design 3. deletedagin 4. deletedal 4. deletedal 4. deletedal 4. deletedal 5. deletedagin 10. error late 10. error late 10			
yge "bad", "copride", "condition AppDotat Lecell Programs Python			
Set of a set			
<pre></pre>			
<pre>insertedmine insertedmine insertedmine</pre>			
<pre>instraid defended defended defended defended defended defended defended defended defended defended defended for values 2 for for defended defended defended for defended de</pre>			
i i i i i i i i i i i i i i i i i i i			
deforming the set of the			
b deferred (A default A default (A default (A default (A default (A default (A default) (A default (A default (A default) (A default (A default) (A default) (A default (A default) (A			
<pre>c determid c determid the run choice exter run choice ext</pre>			
S card water we thole 1 water we have all stater what 3 stater what 3 stater what 3 stater what 3 stater what 3 stater we have 3			
<pre>d upgoy en un otice1 anter un unber of note3 anter un unber of note3 anter unber of note3 anter un otice3 dements in single inked list are: 5 5 5 5 7 6 9 / Phon 12 Shaft anter un otice3 0 9 / Phon 12 Shaft anter un otice3 0 9 / Phon 12 Shaft 5 6 deledeepin 6. deledeepin 6. deledeepin 7. delemant 7. delemant 8. deledeepin 9. despin 9. desp</pre>			
in a constrained of a c			
enter under of node3 enter vule25 enter vule35 enter vule35 55 55 55 55 55 55 55 55 55 55 55 55 5			
enter veluat 5 enter veluat 5 enter veluat 5 enter veluat 5 enter veluat 9 enter veluat 9			
<pre>mire value35 enter ur choice9 enter value35 statu ur choice2 enter ur choice2 enter ur choice3 enter ur choice3 enter ur choice4 * constatu ar choice6 * constatu ar choic</pre>			
enter vulkas enter vulkas en			
enter ur choice elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5			
elements in single linked list are: 15 25 25 25 25 25 25 25 25 25 2			
15 25 25 25 25 25 25 25 25 25 2			
25 35 tarter ur choice2 autor ur choice9 15 25 25 35 autor ur choice9 16 phon 37.3 Stell 26 phon 37.3 Stell 26 phon 37.3 Stell 26 phon 37.3 Stell 26 atter ur choice 26 atter ur choice9 26 deterted 7. detertaid 8. detertaid 9. display 10. exit 10. exit 10. exit 10. exit 10. exit 10. exit 10. exit 11. exit 15. ex			
35 enter ur choice2 enter ur choice3 enter ur choice4 enter ur choice enter ur choice enter ur choice enter ur choice enter ur choice enter ur choice3 enter ur choice3 enter ur choice2 enter ur choice4 enter ur choice5 enter ur choice5 enter ur choice4 enter ur choice5 enter ur choice5 enter ur choice5 enter ur choice4 enter ur choice4 enter ur choice4 enter ur choice5 enter ur choice5 enter ur choice5 enter ur choice5 enter ur choice6 enter ur choice6 enter ur choice6 enter ur choice5 enter ur choice5 enter ur choice6 enter ur choi			
enter un choice? enter un cho			
enter v unels dements in single linked list are: 3 3 3 3 arter ur choice			
and a transformer of the second secon			
structure at angle function of eq. 5 5 5 5 5 5 6 6 6 7 7 7 9 9 9 9 9 9 9 9 10 10 10 10 10 10 10 10 10 10			
1 2 2 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4			
273 353 enter ur choice * Python 3.73 Shelt Sie feit She Debug Options Wordow Help 5. deleteendi 6. deleteendi 7. deleteendi 8. count 9. display 10. exit enter ur choice enter vinle25 enter vinle26 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5			
35 enter ur choice *Python 37.3 Shalf* Ele Edit Shall Debug Options Window Help 5. deleteenda 7. deletemid 8. count 9. display 10. exit enter number of node3 enter value35 enter value35 enter value35 enter value35 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice9 elements in single linked list are: 5 15 25 35 enter ur choice9 elements in single linked list are: 5 16 25 35 enter ur choice9 elements in single linked list are: 5 17 28 38 18 29 29 20 39 elements in single linked list are: 5 19 20 21 23 35 enter ur choice9 elements in single linked list are: 5 21 25 35 enter ur choice9 elements in single linked list are: 5 21 23 35 enter ur choice9 elements in single linked list are: 5 21 23 35 enter ur choice9 elements in single linked list are: 5 24 35 25 35 26 27 36 27 37 37 37 37 37 37 37 37 37 3			
enter ur choice Python 373 Shelf De Liett Shel Debug Options Window Heip S. deletebegin 6. deletendi 8. colutt 9. display 10. exit enter ur choice1 enter value25 enter value25 enter value25 enter value35 enter value36 enter value36			
Python 3.73 Shelf Sole Cet Shell Python 3.73 Shelf Sole Cet Shell Sole Ce			
<pre>% Python 37.3 Shell* is Edit Shell Debug Options Window Help i. deletebegin i. deletebedi /. deletemid /</pre>			
Elie Edit Shell Debug Options Window Help 5. deletebegin 6. deletemid 7. deletemid 8. count 9. display 10. exit enter wichoice1 enter number of node3 enter value15 enter value25 enter value35 enter value35 enter value35 enter value35 enter value5 enter walue5 en	33		đ
5. deletebegin 6. deletebegin 7. deletemid 8. count 9. display 10. exit enter unober of node3 enter unober of node3 enter value25 enter value25 enter value35 enter value35 enter value36 elements in single linked list are: 15 25 35 enter ur choice9 elements in single linked list are: 5 5 16 15 15 15 15 16 16 17 18 19 29 29 20 20 20 21 25 25 25 25 25 25 25 25 25 25			
6. delerend 7. delerend 8. count 9. display 10. exit enter unber of node3 enter value15 enter value25 enter value35 enter value35 enter value35 enter value35 enter value35 enter value35 enter value5 enter value5			
7. delemid 8. count 9. display 10. exit enter unuber of node3 enter value25 enter value25 enter value25 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice2 enter value5 enter ur choice3 enter ur choice4 elements in single linked list are: 5 15 25 25 25 25 25 25 25 25 25 25 25 25 25			
8. count 9. display 10. exit enter une choice 1 enter nuhuer of node3 enter value15 enter value25 enter value25 enter value35 enter value35 enter un choice9 elements in single linked list are: 15 25 35 enter un choice2 enter value5 enter un choice3 enter value45 enter un choice3 enter un choice3 enter un choice3 enter un choice3 enter un choice4 enter un choice5 enter un choice5 enter un choice3 enter un choice4 enter un choice5 enter un cho			
9. display 10. exit enter unbeioe I enter value 25 enter value 25 enter value 25 enter ur choice 9 elements in single linked list are: 15 25 35 enter ur choice 2 enter value 5 enter value 5 enter value 5 s 15 25 35 enter ur choice 3 enter value 45 enter value 45 enter ur choice 3 enter ur choice 4 enter value 45 enter ur choice 3 enter ur choice 3 enter ur choice 3 enter ur choice 4 enter ur choice			
10. exit enter unuber of node3 enter value15 enter value25 enter value35 elements in single linked list are: 15 25 35 enter ur choice2 enter value5 enter ur choice9 elements in single linked list are: 5 15 25 33 enter ur choice3 enter value45 enter ur choice3 enter value45 enter ur choice4 enter ur choice5 enter ur choice5 enter ur choice6 enter ur choice6 ente			
enter under of node3 enter value15 enter value25 enter value35 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice2 enter value5 enter value5 elements in single linked list are: 5 5 5 enter value5 enter value5 enter value5 enter value5 s enter value5 s enter value5 enter value5 s f 15 25 35 enter value4 enter ur choice9 elements in single linked list are: 5 5 5 5 6 15 25 35 8 16 25 35 8 17 25 35 8 17 25 35 26 35 27 35 28 35 29 29 29 29 29 20 20 20 20 20 20 20 20 20 20 20 20 20			
enter value25 enter value25 enter value25 enter value25 other value35 setter ur choice9 elements in single linked list are: 5 enter value5 elements in single linked list are: 5 5 5 5 5 5 5 5 5 6 enter ur choice3 enter ur choice4 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5			
enter value25 enter value25 enter value25 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice2 enter value5 enter value5 15 25 35 enter ur choice3 enter value45 enter value45 enter value45 enter value45 enter value45 enter value45 enter ur choice3 enter ur choice3 enter ur choice3 enter ur choice4 enter ur choice4 enter ur choice5 enter ur choice5 enter ur choice4 enter ur choice5 enter ur choice5 enter ur choice6 enter ur choice6 ente			
enter value35 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice2 enter value5 enter value5 enter value5 s enter value5 5 5 5 5 enter ur choice3 enter value45 enter value45 s enter value45 enter value45 enter value45 s enter va			
elements in single linked list are: 15 25 25 25 25 26 enter w choice2 enter walue5 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5			
elements in single linked list are: 15 25 35 enter ur choice2 enter value5 enter ur choice9 elements in single linked list are: 5 5 5 5 5 5 6 enter ur choice3 enter ur choice3 enter ur choice4 enter ur choice9 elements in single linked list are: 5 5 5 6 elements in single linked list are: 5 5 6 elements in single linked list are: 5 5 6 elements in single linked list are: 5 6 6 6 6 6 6 6 6 6 7 7 7 7 8 7 8 7 8 8 8 8 8 8 8 8 8 8 8 8 8			
15 25 35 enter ur choice2 enter value5 enter value5 elements in single linked list are: 5 15 25 35 enter value45 enter value45 enter value45 elements in single linked list are: 5 5 5 15 15 25 25 25 25 25 25 25 25 25 2			
25 35 anter value5 enter value5 enter value5 5 5 5 5 5 5 5 5 5 5 5 5 5			
55 enter ur choice2 enter value5 elements in single linked list are: 5 15 25 35 enter ur choice3 enter value45 enter ur choice9 elements in single linked list are: 5 15 25 35 45 enter ur choice9 elements in single linked list are: 5 15 25 35 45 enter ur choice9			
enter ur choice2 enter value5 enter value5 soluter ur choice9 elements in single linked list are: 5 15 25 35 enter ur choice3 enter value45 enter value45 elements in single linked list are: 5 15 25 25 25 25 25 25			
enter value5 elements in single linked list are: 5 5 5 5 5 5 5 6 enter value45 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5			
enter ur choice9 elements in single linked list are: 5 15 25 35 enter ur choice3 enter value45 elements in single linked list are: 5 15 15 15 15 15 15 15 15 15 15 15 15 1			
elements in single linked list are: 5 15 15 25 35 enter value45 enter value45 elements in single linked list are: 5 5 15 25 25 25 25 25 25 25 25 25 2			
5 15 25 25 25 26 enter winded 26 26 25 25 25 25 25 25 25 25 25 25			
15 25 35 anter ur choice3 enter value45 enter ur choice9 elements in single linked list are: 5 15 15 15 15 15 15 15 15 15			
23 35 enter ur choice3 enter vu choice3 elements in single linked list are: 5 5 15 25 25 25 25 45 enter ur choice enter ur choice			
oner ur choice3 enter value45 enter ur choice9 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5			
enter valued5 enter valued5 elements in single linked list are: 5 15 25 35 45 enter ur choice			
enter ur choice9 elements in single linked list are: 5 5 25 35 45 enter ur choice			
lements in single linked list are: 5 15 25 35 45 enter ur choice			
5 15 25 35 45 enter ur choice			
15 25 35 45 enter ur choice			
25 35 45 enter ur choice			
35 45 enter ur choice			
45 enter ur choice			
enter ur choice			
			Ln: 42
		11:18	Ln: 42


Python 3.7.3 Shell	- n ×
Eile Edit Shell Debug Options Window Help	
enter number of node5	^
enter value5	
enter value15	
enter value25	
enter value45	
enter ur choice4	
enter value55	
number of nodes=5	
enter ur choice9	
elements in single linked list are:	
5	
15	
25	
35	
45 anter ur abaixa5	
enter ur choice9	
elements in single linked list are:	
15	
25	
35	
45	
enter ur choice6	
enter ur choice9	
15	
55	
25	
35 anter us obeind	
enter ur choice	In:49 Col: 15
A 目 C I I I I I I I I I I I I I I I I I I	x ^R ∧ ∎ ⊄× @ ENG 11:21 ₹
	11 2019
Distance in	
A "Python 3 / 3 Shell"	- n ×
Python 3.7.3 Shell* File Edit Shell Debug Options Window Help	- 0 ×
Ige "Python 5.7.3 shell" File Edit Shell Debug Options Window Help Inter position 3	- a ×
l@ "Pyrton 5.7.3 Shell" File Edit Shell Debug Options Window Help enter position3 number of nodes=5	- a ×
Image: Type of the state of	- a ×
B Paymon 5.1.3 Shell Elle Edit Shell Debug Qptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are:	×
[d] "Yinton 5.1.3 Shell Elle Edit Shell Debug Options Window Help enter position.3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 IS	- ¤ ×
(a) "Yyton 5.7.3 shell" File Edit Shell Debug Options Window Help enter position3 number of nodes=5 entert ur choice9 elements in single linked list are: 5 55	-
(a) "Yyton 5.7.3 Shell" File Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter tur choice9 elements in single linked list are: 5 15 55 25	- a ×
[@ "Yipton 5.7.3 Shell" File Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 15 25 25 25 25 25 25 25 25 25 25 25 25 25	- a ×
Jappinon 5.7.3 Shell Elle Edit Shell Debug Quptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 55 25 35 45 enter ur choice5	X
[a] "Yyton 5.7.3 Shell" [ile [dit Shell Debug Quptions Window Help enter position.3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 15 55 25 35 45 enter ur choice5 enter ur choice5 enter ur choice9	- ¤ ×
(a) "Yyton 5.1.3 shell" Elle Edit Shell Debug Qptions Window Help enter position.3 number of nodes=5 enter tur choice9 elements in single linked list are: 55 25 35 45 enter ur choice5 enter ur choice9 elements in single linked list are:	-
(# 'Pyrion 5.7.3 Shell' [le Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	-
(a) "Symon 3.7.3 Shell" (b) Rebug Quotions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	- a ×
Be Typinon 3.7.3 Shell Elle Edit Shell Debug Quptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 5 5 5 5 5 6 enter ur choice5 enter ur choice6 elements in single linked list are: 15 5	- a ×
(a) "Yynon 5.1.3 shell" File Edit Shell Debug Quptions Window Help enter position.3 number of nodes=5 enter un choice9 elements in single linked list are: 55 25 35 45 enter un choice9 elements in single linked list are: 15 55 25 35 45 enter un choice5 enter un choice5 enter un choice5 state 55 25 35 45	- a ×
(a) "Yynon 5.1.3 shell" File Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter tur choice9 elements in single linked list are: 55 25 35 45 enter ur choice9 elements in single linked list are: 15 25 35 45 enter ur choice5 enter ur choice6 25 45 enter ur choice5 enter ur choice6	-
[a] "Pyrton 5.7.3 Shell" [a] Shell Debug Quitons Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
(a) "Symon 3.7.3 Shell" [le [dit Shell Debug Qptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice5 enter ur choice6	- a ×
(a) "Symon 3.7.3 Shell" [le [dit Shell Debug Quitons Window Help enter position.3 number of nodes=5 enter ur choice9 elements in single linked list are: 5	- a ×
(a) "Yynon 5.1.3 shell" File Edit Shell Debug Qptions Window Help enter position.3 number of nodes=5 enter ur choice9 elements in single linked list are: 55 25 35 45 enter ur choice9 elements in single linked list are: 15 55 26 35 45 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice6 enter ur choice9 elements in single linked list are: 15 55 25	- a ×
(a) "Pyrton 5.1.3 shell" [b] Edit Shell Debug Quptons Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 25 35 45 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice5 enter ur choice4 elements in single linked list are: 15 55 25 35 45 enter ur choice5 enter ur choice6 enter ur choice9 elements in single linked list are: 15 55 25 35 25 35 25 35 25 35 25 35 25 35 25 35 25	
(a) "Yynon 5.1.3 shell" File Edit Shell Debug Qptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 25 35 45 enter ur choice5 enter ur choice6 enter ur choice7 st	- a ×
(a) "Yyton 5.1.3 shell" File Edit Shell Debug Qptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 55 25 35 45 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice6 enter ur choice9 elements in single linked list are: 15 55 25 35 25 35 25 35 25 35 25 35 25 35 26 37 27 38 39 29 <	
(a) "Yynon 5.1.3 shell" File Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 55 25 35 45 enter ur choice6 enter ur choice9 elements in single linked list are: 15 55 25 35 25 35 25 35 25 35 35 35 35 35 35 36 37	- a ×
(a) "Synon 3.1.3 shell" (b) Synon 3.1.3 shell (b) Synon 3.1.3 shell (b) Synon 3.1.3 shell (b) Synon 3.1.3 shell (c) Synon 3	- a ×
(a) "Pyton 5.1.3 shell" File Edit Shell Debug Qptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 25 35 45 enter ur choice5 enter ur choice6 enter ur choice7 enter ur choice6 enter ur choice7 enter ur choice7 enter ur choice7 enter ur choice6 enter ur choice7 enter ur choice7 enter ur choice7 enter ur choice6 enter ur choice7 enter ur choice6 enter ur choice7 enter ur choice8 elements in single linked list a	
Image: Select	
B Type: State Stell Elle Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 25	
(a) "Yynno 3.1.3 shell" File Edit Shel Debug Qptions Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 5 25 35 45 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice5 enter ur choice6 enter ur choice7 enter ur choice7 enter ur choice6 enter ur choice7 enter ur choice8 elements in single linked list are: 15 25 35 25 35 enter ur choice6	- O ×
(a) "Yynon 3.1.3 shell" [Ble Edit Shell Debug Options Window Help enter position3 number of nodes=5 enter ur choice9 elements in single linked list are: 15 55 25 35 45 enter ur choice5 enter ur choice6 enter ur choice6 enter ur choice6 enter ur choice6 enter ur choice7 enter ur choice7 enter ur choice7 enter ur choice8 elements in single linked list are: 15 55 25 35 35 35	
(a) "Pyrion 3.1.3 shell" File Edit Shell Debug Options Window Help enter variable of nodes=5 enter ur choice9 elements in single linked list are: 5 55 25 35 45 enter ur choice6 enter ur choice7 enter ur choice7 enter of nodes=4 enter ur choice9 elements in single linked list are: 15 55 25 35 25 35 enter ur choice7 enter ur choice9 elements in single linked list are: 15 55 25 35 enter ur choice6 enter ur choice7 enter ur choice7 enter of nodes=4 enter ur choice9 elements in single linked list are: 15 55 25 35 enter ur choice9 elements in single linked list are: 15 55 25 35 enter ur choice7 enter of nodes=4 enter ur choice9 elements in single linked list are: 15 25 35 enter ur choice9 elements in single linked list are: 15 55 25 35 enter ur choice9 elements in single linked list are: 15 55 26 35 enter ur choice9 elements in single linked list are: 15 55 26 35 enter ur choice9 elements in single linked list are: 15 26 27 35 35 36 37 38 38 39 39 30 30 30 31 32 33 35 35 35 36 36 37 37 38 38 39 39 30 30 30 30 31 32 33 34 35 35 35 36 36 36 37 38 38 39 30 3	- □ × • • • • • • • • • • • • • • • • • • •

PRE-LAB VIVA QUESTIONS:

- a. What is linkedlist?
- b. What type of memory allocation is used in linkedlist?
- $c. \ How many self referential pointers are used in single linked list?$
- d. What is double linked list?
- e. Which node contains NULL pointer in a single linkedlist?
- f. How many nodes you can have in a single linkedlist?
- g. What are the components of a polynomial expression?

LAB ASSIGNMENT:

- a. Formulate a program to create a singly linked list and perform insertion, deletion andtraversing operations on a singly linked list.
- b. Write a program to merge two linkedlist?
- c. Compose a program to print odd nodes of a linkedlist?

- d. Write a program to divide the linked list into two parts into odd and evenlist?
- e. Formulate a program to convert a single linked to circular linkedlist?
- $f. \ \ Compose a program to store and add two polynomial expressions in memory using linked list.$

POST-LAB VIVAQUESTIONS:

- a. What is the time complexity to insert a node at the beginning oflinkedlist?
- b. What is the time complexity to traverse a linkedlist?
- c. Howmanymodificationsarerequiredtodeleteanode atthebeginning?
- d. Howmanymodificationsarerequiredtoinsertanode inthemiddle ofthelinkedlist?
- e. What are the types of linkedlist?
- f. What are the applications of a linkedlist?

WEEK – 7

IMPLEMENTATION OF CIRCULAR LINKED LIST

OBJECTIVE:

Write Python script for the following operations on Circular Linked List. (i) Creation (ii) insertion (iii) deletion (iv)traversal

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Circular Linked List:

(i) Creation Init_circular_linked_list(key) z= new node z.data=key z.next=z c=new circular_linked_list c.last=z returnc

(ii) Insertion

Insert_after(n,a) n.next=a.next a.next=n insert_at_last(L,n)

n.next=L.last.next L.last.next=n L.last=n

(iii) Deletion

Delete(L,n)

temp=L.last while temp.next!=n

temp=temp.next

if n==L.last

```
if n.next==n
```

L.last=NULL

else

temp.next=n.next

L.last=temp

else

temp.next=n.next

(iv) Traversal

Node temp = this.last; print temp.data temp = temp.next;

while(temp != this.last) {
 print temp.data
 temp = temp.next;

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.

b. Execute : Go to Run ->Run module(F5)

SOURCE CODE: class Node: def_init_(self,data): self.next=None self.data=data print("Nodecreated",data) class CLList: definit(self): self.head=None self.ctr=0 def insert_beg(self,data): node=Node(data) if self.head==None: self.head=node node.next=self.head else: temp=self.head while temp.next is not self.head: temp=temp.next temp.next=node node.next=self.head self.head=node print("Node inserted",data) self.ctr+=1 return def insert_end(self,data): node=Node(data) if self.head==None: self.head=node node.next=self.head else: temp=self.head while temp.next is not self.head: temp=temp.next temp.next=node node.next=self.head self.ctr+=1 print("Node inserted",data) return def insert_inter(self,pos,data): node=Node(data) if pos<1 or pos>self.ctr: print("invalid position") else: temp=self.head i=1 while i<pos:

temp=temp.next i+=1node.next=temp.next temp.next=node self.ctr+=1 print("Node Insered",data) return def delete_beg(self): ifself.head==None: print("No Nodes exist") elif self.ctr==1: print("Node deleted",self.head.data) self.head=None self.ctr-=1 else: print("Node deleted",self.head.data) temp=self.head while temp.next is not self.head: temp=temp.next self.head=self.head.next temp.next=self.head self.ctr-=1 return def delete_end(self): ifself.head==None: print("No Nodes exist") elif self.ctr==1: print("Node deleted",self.head.data) self.head=None self.ctr-=1 else: temp=self.head prev=temp while temp.next is not self.head: prev=temp temp=temp.next print("Node deleted",temp.data) prev.next=temp.next self.ctr-=1 return def delete_inter(self,pos): if self.head==None: print("No nodes exist") elif pos<1 or pos>self.ctr: print("Invalid position") elif self.ctr==1: print("Node deleted",self.head.data) self.head=None self.ctr-=1 else: temp=self.head prev=temp

i=0while i<pos: prev=temp temp=temp.next i+=1prev.next=temp.next print("Node deleted",temp.data) self.ctr-=1 return def traverse(self): temp=self.head i=0while i<self.ctr: print(temp.data) temp=temp.next i+=1 return def Menu(): print("1.Insert at beginning") print("2.Insert at middle") print("3.Insert at end") print("4.Delete at beginning") print("5.Delete at middle") print("6.Delete at end") print("7.Traverse Forward") print("8.Number of nodes") print("9.Exit") ch=int(input("Enter choice:")) returnch c=CLList() whileTrue: ch=Menu() ifch==1: data=input("Enter data:") c.insert_beg(data) elif ch==2: data=input("Enter data:") pos=int(input("Enter position:")) c.insert_inter(pos,data) elif ch==3: data=input("Enter data:") c.insert_end(data) elif ch==4: c.delete beg() elif ch==5: pos=int(input("Enter position:")) c.delete_inter(pos) elif ch==6: c.delete_end()

```
elifch==7:
c.traverse()
elifch==8:
print("Number of Nodes",c.ctr)
else:
print("Exit")
break
```

Output:



ile Edit Shell Debug Ontions Window Help	- 0 ×
The For Such Scholl Shung Winger	
Delete at middle	·
5.Delete at end	
/.Traverse Forward	
8.Number of nodes	
.Exit	
Enter choice:2	
Inter data:45	
Inter position 2	
Vode created 45	
Vode Insered 45	
Insert at beginning	
Insert at middle	
Insert at end	
Delete at beginning	
Delete at middle	
Delete at end	
Traverse Forward	
Number of nodes	
Exit	
Inter choice 7	
5	
5	
5	
5	
Juncent at basinning	
Insert at beginning	
Insert at induce	
Insert at end	
Delete at beginning	
Delete at middle	
"Delete at end	
.Iraverse Forward	
Number of nodes	
Exit	
Enter choice:	×
	Ln: 93 Col: 1
	-8 A T 44 G DIG 10:52
	X
3 Turken 2 7 2 Challs	
• "Python s.r.s sheir"	- 0 ^
ile Edit Shell Debug Options Window Help	
.Delete at middle	· · · · · · · · · · · · · · · · · · ·
Delete at end	
Traverse Forward	
Number of nodes	
Exit	
nter choice:3	
nter choice:3	
nter choice: 3 nter data: 55 fold created 55	
nter choice:3 nter data:55 ode created 55 ode inserted 55	
Inter choice: 3 Inter choice: 3 Inter choice: 3 Inter choice: 5 Inter choice:	
nter choice: 3 nter data;55 ode created 55 tode inserted 55 Insert at beginning Insert at beginning	
Inter choice: 3 Inter choice: 3 Inter choice: 3 Inter choice: 5 Inter choice:	
Inter choice: 3 forde created 55 forde created 55 loade inserted 55 Insert at beginning Insert at end Delata at beginning	
Inter choice: 3 Inter choice: 3 Inter choice: 3 Inter chai: 55 Inter chai: 55 Insert at beginning Insert at middle Insert at middle Delete at beginning Delete at beginning	
Inter choice: 3 inter choice: 3 inter choice: 3 inter choice: 3 inter choice: 4 inter choice:	
Inter choice: 3 inter choice:	
Inter choice: 3 inter choice: 3 inter choice: 3 inter choice: 3 inter choice: 4 inter choice:	
htter choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at end Delete at beginning Delete at end Delete at end Traverse Forward Number of nodes	
ther choice:3 ode created 55 ode inserted 55 Insert at middle Insert at middle Delete at beginning Delete at middle Delete at middle Delete at middle Delete at middle Delete at middle Exit	
hter choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at middle Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Number of nodes Exit hter choice:7	
htter choice:3 ode created 55 ode inserted 55 Insert at eginning Delete at middle Delete at beginning Delete at niddle Delete at end Traverse Forward Number of nodes Exit Insert at end Delete 35 Delete 30 Delete 30 D	
ther choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at middle Delete at middle Delete at middle Delete at middle Delete at middle Delete at middle Delete at middle Exit Exit atter choice:7 5	
ther choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at middle Delete at beginning Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit ther choice:7 5	
ther choice:3 ode created 55 ode inserted 55 Insert at middle Insert at middle Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Number of nodes Exit ther choice:7 5	
htter choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at middle Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Number of nodes Exit nter choice:7 5 5	
ter choice:3 ode created 55 ode inserted 55 Insert at middle Insert at middle Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Number of nodes Exit ter choice:7 5 5 5 6 6 7	
ther choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at middle Delete at middle Delete at middle Delete at middle Delete at middle Delete at middle Exit Insert at beginning Insert at beginning Insert at beginning Insert at beginning Insert at beginning	
htter choice: 3 ode created 55 ode created 55 Insert at 65 Insert at spinning Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Number of nodes Exit hter choice: 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
htter choice: 3 ode created 55 ode created 55 Insert at beginning Insert at middle Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit htter choice: 7 5 5 5 5 5 5 5 5 5 5 5 5 5	
nter choice:3 fode created 55 fode created 55 Insert at segnining Insert at middle Insert at middle Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit nter choice:7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
inter choice:3 folde created 55 lode inserted 55 Insert at beginning Insert at middle Insert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit inter choice:7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
nter choice:3 fode created 55 fode created 55 Insert at beginning Insert at middle Insert at end Delete at middle Delete at end Traverse Forward Number of nodes Exit nter choice:7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
inter choice:3 Jode created 35 Jode created 35 Jode inserted 55 Insert at beginning Delete at middle Delete at middle Delete at middle Delete at end Traverse Forward Number of nodes Exit Inter choice:7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
inter choice:3 iode created 55 iode created 55 Iode inserted 55 Insert at beginning Insert at middle Delete at middle Delete at middle Delete at middle Exit inter choice:7 5 5 5 5 5 5 5 5 5 5 5 5 5	
inter choice:3 lode created 55 lode created 55 lode inserted 55 lnsert at beginning Delete at middle Delete at middle Delete at middle Traverse Forward Number of nodes Exit inter choice:7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	
niter choice: 3 folde created 55 folde insert at middle Delete at middle Delete at middle Delete at middle Traverse Forward For the semining F	
ther choice:3 ode created 55 ode inserted 55 Insert at beginning Insert at middle Insert at middle Delete at end Traverse Forward Number of nodes Exit atter choice:7 \$ 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	Lr: 123 Col: 1

B *Python 3.7.3 Shell*	- 0 ×
File Edit Shelj Debug Options Window Help	
2.Insert at middle	^
3.Insert at end	
4. Delete at beginning	
5.Delete at and	
2. Traves Forward	
8.Number of nodes	
9.Exit	
Enter choice:4	
Node deleted 35	
1.Insert at beginning	
2. Inset a middle	
J Delete at baginning	
5.Delete at middle	
6.Delete at end	
7.Traverse Forward	
8. Number of nodes	
9.Extr abian?	
25	
45	
15	
55	
1.Insert at beginning	
2.Insett at middle	
3. Insert at end	
5. Delete at middle	
6.Delete at end	
7.Traverse Forward	
8.Number of nodes	
9.Exit	
Enter choices	In: 148 Col: 13
	10:53 E
	⊄× // ₆ ENG 10:53 ■
Python 3.7.3 Shell* Elle Edit Shell Debug Options Window Help	¢× 🦟 ENG 10:53 📮
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning	¢× <u>@</u> ENG 1053 ₽ 11 Jul 2019 ₽ -
Python 3.7.3 Shell ⁴ Elle Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at middle	¢× <u>@</u> ENG 1053 ₽ 11 Jul 2019 ₽ -
Python 3.7.3 Shell Elie Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at middle 3.Insert at end	4× <u> </u>
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at end 4.Delete at beginning 5.Delete at windle	¢× <u> </u>
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at middle 3.Insert at end 4.Delete at beginning 5.Delete at middle 6 Delete at end	¢× <u> </u>
Python 3.7.3 Shell* Insert at beginning S.Delete at middle S.Delete at mid	¢× <u> </u>
Peter and Applied	4× <u> </u>
P II P III P II	¢× <u> </u>
Python 3.7.3 Shell* File Edit Shell Debug Options Window Help I.Insert at beginning Z.Insert at middle S.Insert at niddle S.Insert at	¢× <u> </u>
Python 3.7.3 Shell* In Section 2.1.3 Shell* In Section 2.1.	4× <u><u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u></u></u>
Period 2.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning S.Insert at end 4.Delete at beginning S.Delete at middle 6.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice.6 Node deleted 55 I.Insert at beginning Z.Insert at middle	4× <u><u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u><u></u></u></u>
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at middle 3.Delete at middle 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 I.Insert at middle 3.Insert at end 3.Insert at end	¢× <u><i>i</i></u> , ENG 1053 11 Jul 2019 ♥ -
Python 3.73 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning S.Delete at middle 3.Insert at hed 4. Delete at beginning 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 I.Insert at beginning 2.Insert at middle 3.Insert at hedining 2.Insert at middle 3.Insert at beginning 4.Delete at beginning 4.Delete at beginning	¢× <u><i>i</i></u> , <u>ENG</u> 11 <u>Jul</u> 2019 −
Python 3.7.3 Shell* In Edit Shell Debug Options Window Help Insert at beginning S.Delete at middle S.Traverse Forward S.Number of nodes S.Exit Enter choice:6 Node deleted 55 I.Insert at beginning S.Insert at middle S.Insert	4× <u><i>i</i></u> <u>k</u> ENG 1153 11 Jul 2019 −
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning S.Insert at end 4.Delete at beginning S.Delete at middle 6.Delete at middle S.Number of nodes 9.Exit Enter choice.6 Node deleted 55 I.Insert at beginning S.Insert at end 4.Delete at end S.Insert at end 4.Delete at middle S.Insert at middle S.Delete at middle S.Delete at middle S.Delete at middle S.Insert at middle S.Insert at middle S.Delete at Middle	4× <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i><u>i</u><u><i>i</i></u><u><i>i</i><u></u><u><i>i</i></u><u><i>i</i><u></u><u><i>i</i></u><u><i>i</i></u></u></u></u>
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at end 4.Delete at beginning 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 1.Insert at end 4.Delete at end 7.Traverse Forward 8.Insert at end 4.Delete at end 7.Traverse Forward 4.Delete at end 7.Traverse Forward 4.Delete at end 7.Traverse Forward	¢× <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i><u>i</u><u><i>i</i></u><u><i>i</i><u></u><u><i>i</i></u><u><i>i</i></u></u></u>
Python 3.73 Shell* Elle Edit Shell Debug Options Window Help 1.Insert at beginning 2.Insert at middle 3.Insert at ned 4.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 1.Insert at beginning 2.Insert at middle 3.Insert at hed 4.Delete at middle 3.Delete at hed 4.Delete at hed 4	4× <u><i>i</i>k</u> ENG 1153 11 Jul 2019 →
Poption 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.nsert at middle 3.nsert at end 4.Delete at beginning 5.Delete at middle 6.Delete at middle 6.Delete at middle 5.List Enter choice:6 Node deleted 55 I.nsert at middle 3.Insert at end 4.Delete at middle 5.Delete at middle 5.Delete at middle 7.Traverse Forward 8.Insert at end 4.Delete at middle 5.Delete at middle 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:7	4× <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i><u>i</u><u><i>i</i></u><u><i>i</i><u><i>i</i></u><u><i>i</i></u><u><i>i</i></u></u></u>
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning S.Insert at end 4.Delete at beginning S.Delete at middle 6.Delete at middle 6.Delete at middle S.Number of nodes 9.Exit Enter choice.6 Node deleted 55 I.Insert at beginning S.Delete at middle 3.Insert at end 4.Delete at end 7.Traverse Forward 8.Sumber of nodes 9.Exit Enter choice.6 Node deleted 55 I.Insert at middle 3.Insert at end 4.Delete at end 7.Traverse Forward 8.Sumber of nodes 9.Exit Enter choice.6 Node deleted 55 I.Insert at middle 3.Insert at end 4.Delete at end 7.Traverse Forward 8.Sumber of nodes 9.Exit Enter choice.7 25	4× <u><i>R</i></u> ENG 1153 11 Jul 2019 −
Python 3.7.3 Shell* Ele Edit Shell Debug Options Window Help I.Insert at beginning Z.Insert at middle 3.Insert at ed 4.Delete at beginning 5.Delete at middle 6.Delete at end T.Travesse Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 1.Insert at beginning 2.Insert at middle 3.Insert	¢× <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i><u>i</u><u><i>i</i></u><u><i>i</i><u><i>i</i></u><u><i>i</i></u><u><i>i</i></u></u></u>
Python 3.73 Shell* Elle Edit Shell Debug Options Window Help 1.Insert at beginning 2.Linsert at middle 3.Insert at end 4.Delete at middle 6.Delete at middle 6.Delete at middle 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 1.Insert at beginning 5.Delete at middle 3.Insert at edit 4.Delete at beginning 5.Delete at middle 6.Delete at middle 6.Delete at middle 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:7 25 45 15	4× <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i><u>i</u><u><i>i</i></u><u><i>i</i><u><i>i</i></u><u><i>i</i></u><u><i>i</i></u></u></u>
Python 3.7.3 Shelf File Edit Shelf Debug Options Window Help 1. Insert at beginning 2.Insert at middle 3.Insert at end 4. Delete at middle 6. Delete at middle 6. Delete at end 7. Traverse Forward 8. Number of nodes 9. Exit Enter choice:6 Node deleted 55 1. Insert at beginning 5. Delete at end 7. Traverse Forward 8. Number of nodes 9. Exit Enter choice:6 Node deleted 55 1. Insert at beginning 5. Delete at end 7. Traverse Forward 8. Number of nodes 9. Exit Enter choice:7 25 45 15 11. Insert at beginning 21. Exit Enter choice:7 25 15 15 15 15 15 15. Insert at beginning 21. Insert at beginning 21. Insert at beginning	4× <u><i>R</i></u> ENG 1153 −
Python 37.3 Shell* File Edit Shell Debug Options Window Help I.Insert at beginning 2.Insert at middle 3.Insert at beginning 5.Delete at middle 6.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 55 1.Insert at beginning 2.Insert at middle 3.Insert at beginning 5.Delete at end 4.Delete at beginning 5.Delete at end 4.Delete at edit 4.Delete at beginning 5.Delete at middle 6.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit Enter choice:7 25 45 15 1.Insert at beginning 2.Insert at middle 3.Insert at middle	
Python 3.7.3 Shelt Fle Edit Shell Debug Options Window Help I.insert at beginning 2.insert at middle 3.insert at end 4.Delete at beginning 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice:6 Node deleted 35 1.insert at beginning 5.Delete at middle 6.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit Enter choice:7 25 45 15 1.insert at beginning 2.insert at middle 3.insert at middle 3.insert at middle 3.insert at beginning 1.insert at beginning 1.insert at beginning 3.insert at middle 6.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit 1.insert at beginning 5.Delete at middle 6.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit 1.insert at beginning 5.Delete at middle 6.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit 1.insert at beginning 3.insert at end 4.Delete at beginning 5.Delete at middle 5.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit 1.insert at beginning 3.insert at end 4.Delete at beginning 5.Delete at middle 5.Delete at end 7.Travese Forward 8.Number of nodes 9.Exit 1.insert at beginning 4.Delete at beginning 5.Delete at middle 5.Delete at end 5.Delete	4× <u><i>i</i>k</u> ENG 1153 11 Jul 2019 →
Python 3.7.3 Shell* JBe Edit Shell Debug Options Window Belp 1.Insert at beginning 2.Insert at middle 3.Insert at end 4.Delete at beginning 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice 6 Node deleted 35 1.Insert at beginning 2.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit 1.Insert at beginning 2.Insert at middle 3.Delete at middle 6.Delete at middle 3.Insert at end 4.Delete at beginning 5.Delete at middle 6.Delete at end 7.Traverse Forward 8.Number of nodes 9.Exit Enter choice? 25 15 1.Insert at beginning 2.Insert at middle 3.Insert at end 4.Delete at beginning 3.Insert at end 4.Delete at beginning 3.Insert at end	4× <u><i>i</i></u> <u><i>i</i></u> <u><i>i</i><u>i</u><u><i>i</i></u><u><i>i</i><u><i>i</i></u><u><i>i</i></u><u><i>i</i></u></u></u>
Provide and a second s	
Python 3.7.3 Shelf Fie Edit Shelf Fie Add Debug Options Window Help I.Insert at beginning I.nsert at middle S.Delete at middle Oblete at end Pistit Enter choice:6 Node deleted 55 I.nsert at beginning S.Delete at middle Oblete at end ADelete at beginning S.Delete at middle Oblete at end ADelete at beginning S.Delete at middle I.nsert at beginning I.nsert at beginning S.Delete at middle I.nsert at beginning S.Number of nodes S.Sett Insert at beginning S.Delete at middle I.nsert at beginning S.Delete at middle S.Delete at middle I.nsert at beginning S.Delete at middle S.	
Python 3.73 Shelf File Edit Shell Debug Options Window Help Insert at beginning Insert at end ADelete at beginning Subset at middle Sublete at end ADelete at beginning Sublete at end Traverse Forward Number of nodes PExit Enter choicec6 Node deleted 55 Insert at beginning Insert at beginning Subset at middle Subset at middle Subset at middle Subset at end ADelete at beginning Subset at end Insert at beginning Subset at middle Subset at middle Subset at middle Subset at end Insert at beginning Insert at beginning Subset at end Traverse Forward Sumber of nodes Subset at end Insert at beginning Subset at end Traverse Forward Sumber of nodes PExit Enter choice:7 Z5 IS Insert at end ADelete at beginning Subset at end Insert at middle Subset at end ADelete at beginning Subset at end Insert at middle Subset at end Insert at middle Subset at end ADelete at end Traverse Forward Sumber of nodes PExit	4× <u>m</u> eng 1153 11 Jul 2019 ₽ -

= 오 밝 😂 🛃 🔵 🕅

א^פ ^ ₪ ⊄× 🦟 ENG 11 Jul 201

Ln: 172 Col: 13

=

Python 3.7.3 Shell	- a ×
.le <u>E</u> dit She <u>l</u> l <u>D</u> ebug <u>O</u> ptions <u>W</u> indow <u>H</u> elp	
Insert at beginning	^
Insert at middle	
Delete at beginning	
Delete at middle	
Delete at end	
Traverse Forward	
Number of nodes	
Exit	
ner enoice.5	
ode deleted 15	
Insert at beginning	
Insert at middle	
Insert at end	
Delete at beginning	
Delete at end	
Traverse Forward	
Number of nodes	
Exit	
ater choice:7	
Insert at beginning	
Insert at middle	
Insert at end	
Delete at beginning	
Delete at middle	
Delete at end	
I raverse Forward	
Exit	
nter choice:	
	Ln: 196 Col: 13
🔹 o h: 🔿 🖪 🖿 🙈 📠	-8 ∧ E1 4x € ENG ^{10:53} ■
•	X* // UNA /// 11 Jul 2019
Python 3.7.3 Shell	- a ×
e <u>E</u> dit Shell <u>Debug Options Window H</u> elp	
Insert at middle	
Insert at end	
Insert at end Delete at beginning	
Insert at end Delete at beginning Delete at middle	
Insert at end Delete at beginning Delete at end Delete at end	
insert at end Delete at beginning Delete at iniddle Delete at end Traverse Forward	
insert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Fvit	
insert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit ter choice:7	
insert at end Delete at beginning Delete at middle Delete at end Fraverse Forward Number of nodes Exit ter choice:7	
insert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit ter choice:7	
insert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit ter choice:7	
nsert at end Delete at beginning Delete at middle Delete at end Fraverse Forward Number of nodes Sxit ter choice:7 nsert at middle	
nsert at end Delete at middle Delete at middle Delete at end Fraverse Forward Swit ter choice:7 nsert at beginning nsert at end Delete at beginning	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit et choice:7 nsert at middle nsert at middle Delete at beginning Delete at beginning	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Sxit ter choice:7 nsert at middle nsert at middle Delete at beginning Delete at traveling Delete at middle Delete at middle	
nsert at end Delete at beginning Delete at middle Delete at middle Swits ter choice:7 nsert at beginning nsert at middle nsert at end Delete at beginning Delete at tend Delete at end Traverse Forward	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:7 nsert at beginning nsert at middle nsert at end Delete at middle Delete at end Traverse Forward Sumber of nodes	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Smite ter choice:7 nsert at middle nsert at middle Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Smite	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Sxit ter choice:7 nsert at beginning nsert at middle Delete at beginning Delete at beginning Delete at middle Delete at end Traverse Forward Sumber of nodes Sxit ter choice:8	
nsert at end Delete at middle Delete at middle Delete at middle Switer choice:7 Insert at beginning Insert at middle Delete at beginning Delete at beginning Delete at middle Delete at middle D	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:7 nsert at middle nsert at middle Delete at beginning Delete at end Delete at beginning Delete at end Delete at end Delete at end Exit Traverse Forward Sumber of nodes Exit ter choice:8 mber of Nodes 2 nsert at beginning nsert at modele	
nsert at end Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Sxit ter choice:7	
nsert at end Delete at middle Delete at middle Delete at end Traverse Forward Aumber of nodes Exit ter choice: 7 Insert at beginning nsert at middle Delete at beginning Delete at beginning Delete at middle Delete at normal Traverse Forward Simber of nodes Exit ter choice: 8 mber of Nodes 2 nsert at end Delete at end msert at end Delete at beginning nsert at middle Delete at beginning nsert at middle Delete at beginning nsert at middle Delete at beginning nsert at middle	
nsert at end Delete at middle Delete at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:7 nsert at middle nsert at middle Delete at middle	
nsert at end Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Sxit ter choice:7 nsert at middle nsert at middle Delete at beginning Delete at beginning Delete at middle Delete at end Traverse Forward Sumber of nodes Sxit ter choice:8 mber of Nodes 2 nsert at middle Delete at middle	
nsert at end Delete at middle Delete at middle Delete at end Traverse Forward Swit ter choice:7 Insert at beginning nsert at middle Delete at end Delete at end Delete at end Delete at end Eraverse Forward Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Swit ter choice:8 Delete at end Delete at end	
Insert at end Delete at beginning Delete at middle Delete at end Traverse Forward Number of nodes Exit ter choice:7 Insert at beginning Delete at middle Insert at middle Delete at end Delete at end Delete at end Traverse Forward Number of nodes Exit ter choice:8 miber of Nodes 2 Insert at end Delete at end Insert at end Delete at middle Insert at end Delete at middle Delete at end Traverse Forward Number of nodes 2 Insert at middle Delete at end Delete at middle Insert at end Delete at end Delete at middle Delete at end Traverse Forward Number of nodes Exit	
Insert at end Delete at middle Delete at middle Delete at middle Traverse Forward Number of nodes Exit ter choice:7 Insert at beginning Insert at middle Insert at middle Delete at beginning Delete at beginning Delete at deginning Delete at middle Delete at middle Insert at not Delete at middle Delete at middle	
insert at end Delete at middle Delete at middle Delete at middle Delete at middle Sxit ter choice:7 insert at middle insert at middle Delete at beginning Delete at beginning Delete at middle Delete at middle Sxit ter choice:8 it middle Delete at middle Delete	
nsert at end Delete at beginning Delete at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:7 msert at beginning nsert at end Delete at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:8 mber of Nodes 2 nsert at beginning nsert at end Delete at delete msert at end Delete at beginning nsert at middle nsert at end Delete at beginning nsert at middle nsert at end Delete at beginning nsert at middle nsert at end Delete at beginning Delete at beginning State to poice:8 mber of Nodes 2 nsert at beginning nsert at middle nsert at end Delete at beginning Delete at beginning Delete at tod Traverse Forward Sumber of nodes Exit ter choice:9 it	
nsert at end Delete at beginning Delete at middle Delete at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:7 nsert at middle nsert at middle Delete at beginning Delete at end Traverse Forward Sumber of nodes Exit ter choice:8 mber of Nodes 2 nsert at beginning nsert at middle nsert at middle Delete at end Traverse Forward Sumber of nodes Exit ter choice:9 it >	Lr: 209 Cot.
<pre>sert at end belete at middle belete at middle belete at middle shareves Forward aumber of nodes xit ter choice:7 sert at middle selete at middle belete at beginning belete at middle belete at middle belete at nod wumber of nodes 2 isert at middle belete at beginning isert at middle belete at beginning isert at middle belete at beginning isert at middle belete at nod belete at nod belet</pre>	Lr: 209 Cet
<pre>sert at end velete at middle velete at middle velete at middle vumber of nodes xit er choice:7 sert at beginning veset at middle velete at beginning velete at middle velete at beginning velete at middle velete at middle velete at nod verses Forward umber of nodes 2 sert at middle velete at velete at middle velete at middle velete at velete at velete velete at velete at velete velete at velete at velete velete at veletevelete velete at veletevele</pre>	Ln: 209 Col.× x ^A ^ IC 4× //, ENG 1054 11 Jul 209 ■

- a. What is circular linkedlist?
- b. What type of memory allocation is used in linked circular list?
- c. How many self referential pointers are used in circular single linkedlist?
- d. What is double linkedlist?
- e. Which node contains NULL pointer in a circular single linked list?
- f. How many nodes you can have in a circular single linkedlist?

LAB ASSIGNMENT:

- a. Formulate a program to create a circular singly linked list and perform insertion, deletionand traversing operations on a singly linkedlist.
- b. Write a program to merge two linkedlist?

- c. Compose a program to print odd nodes of a circular linkedlist?
- d. Write a program to divide the circular linked list into two parts into odd and evenlist?
- e. Formulate a program to convert a single linked to circular linkedlist?

POST-LAB VIVA QUESTIONS:

- a. What is the time complexity to insert a node at the beginning of circularlinkedlist?
- b. What is the time complexity to traverse a circular linkedlist?
- $c. \hspace{0.1 cm} How many modifications are required to delete an ode at the beginning?$
- d. How many modifications are required to insert a node in themiddle of the circular linkedlist?
- e. What are the types of linkedlist?
- f. What are the applications of a circular linkedlist?

WEEK-8

IMPLEMENATION OF DOUBLE LIKED LIST

OBJECTIVE:

Write Python programs for the following operations on Double Linked List. (i) Creation (ii) insertion (iii) deletion (iv) traversal in bothways.

RESOURCE:

Python 3.7.3

PROGRAMLOGIC:

- Double Linked List
- (i) Creation
- (ii) Insertion
- (iii) Deletion
- (iv) Traversal in bothways

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.

b. Execute : Go to Run -> Run module(F5)

SOURCE CODE:

class Node: def_init_(self,data): self.data=data self.next=self.prev=None

class DLinkedList: def init (self): self.head=None self.ctr=0 def insert_beg(self,data): node=Node(data) if self.head==None: self.head=node else: node.next=self.head self.head.prev=node self.head=node self.ctr +=1 print("Nodes inserted",data) return def insert_end(self,data): node=Node(data) if self.head==None: self.head=node else: temp=self.head while(temp.next is not None): temp=temp.next temp.next=node node.prev=temp self.ctr +=1 print("Node inserted",data) return def delete_beg(self): ifself.head==None:

print("No node exist") else: print("Node deleted",self.head.data) self.head=self.head.next self.head.prev=None self.ctr -=1 return def delete_end(self): ifself.head==None: print("No nodes exist") elif self.ctr==1: self.ctr=0 print ("Node deleted",self.head.data) self.head=None else: temp=self.head while temp.next is not None: temp=temp.next print("Node deleted",temp.data) temp=temp.prev temp.next=None self.ctr -=1 return def insert_pos(self,pos,data): if pos==0: self.insert_beg(data) elif pos==self.ctr: self.insert_end(data) else: node=Node(data) temp=self.head i=1 while i<pos-1: temp=temp.next i +=1 node.next=temp.next temp.next.prev=node temp.next=node node.prev=temp self.ctr +=1 print("Node inserted",data) return def delete_pos(self,pos): if self.head==None: print("Node is empty") else: if pos==0: self.delete_beg() elif pos==self.ctr: self.delete_end() else: temp=self.head i=0while i<pos: temp=temp.next i+=1 print("node deleted",temp.data) temp.prev.next=temp.next temp.next.prev=temp.prev temp.next=None

temp.preve=None self.ctr -=1 return def traverse f(self): if self.head==None: print("No nodes exist") temp=self.head i=0while i<self.ctr: print(temp.data) temp=temp.next i+=1 return def traverse_r(self): if self.head==None: print("No nodes exist") temp=self.head while temp.next is not None: temp=temp.next while temp is not None: print(temp.data) temp=temp.prev defmenu(): print("1.Insert at beginning") print("2.Insert at position") print("3.Insert at end") print("4.Delete at beginning") print("5.Delete at position") print("6.Delete at end") print("7.Count no of nodes") print("8.Traverse forward") print("9.Traverse reverse") print("10.Quit") ch=eval(input("Enter choice:")) returnch d=DLinkedList() while True : ch=menu() if ch == 1: data=eval(input("Enter data:")) d.insert_beg(data) elif ch==2: data=eval(input("Enter data:")) pos=int(input("Enter position:")) d.insert_pos(pos,data) elif ch==3: data=eval(input("Enter data:")) d.insert_end(data) elif ch==4: d.delete_beg() elif ch==5: pos=int(input("Enter position:")) d.delete_pos(pos) elif ch==6: d.delete_end() elifch==7: print("Number of nodes",d.ctr) elifch==8:

```
d.traverse f()
elif ch==9:
  d.traverse_r()
else:
  print("Exit")
  break
```

Output:



🛃 *Python 3.7.3 Shell*	- a ×
Elle Edit Shell Debug Options Window Help	
6.Delete at end	
8 Traverse forward	
9. Traverse reverse	
10.Quit	
Enter choice:8	
35	
25	
15 1 Insert at beginning	
2.Insert at position	
3.Insert at end	
4.Delete at beginning	
5.Delete at position	
6.Delete at end	
8. Traverse forward	
9. Traverse reverse	
10.Quit	
Enter choice:9	
15	
35	
1.Insert at beginning	
2.Insert at position	
3.Insert at end	
4.Delete at beginning	
5.Delete at position	
6.Delete at end	
8. Traverse forward	
9. Traverse reverse	
10.Quit	
Enter choice:	v
	Ln: 83 Col: 13
- 🌐 🔎 🗎 🤮 📄 🧔 🕎 🔄	x ^R ∧ IED ⊈× //2 ENG 11 Jul 2019 📮
Puthon 3.7.3 Shell	– n ×
Eile Felt Shall Dahua Ontione Window Halo	and the second se
A REAL PROPERTY AND A REAL	
The For she body throw the	^
The form the of the second sec	^
The first step geody sports minute field The first step geody sports first step geody sports for the first step geody step geod	^
Lee Lee area area Leevag School Zindow Lieup Count no of nodes S.Traverse forward 9.Traverse reverse 10.Quit	^
T.Count no of nodes T.Count no of nodes T.Taverse forward T.Count no of nodes T.Taverse forward T.Count T.Taverse reverse T.Taver	î
7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter data:2	Â
7.Court no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter position:2 Node insected 2	^
Lee Lee Lee area Leevag Agnosto Entrow Teep 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter choice:2 Enter position:2 Node inserted 2 Unsert at becompting	Â
The first are grown and grown	
Dec Lord Step Level approve Endow Tepp 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter choice:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at position 3.Insert at end	Â
Der Lein Ange Zeung Agnetis Ennow Teip 7.Comt no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at position 3.Insert at beginning	^
Dec Len Seig Level genetic gen	
Det gene area general genera general general general general general ge	
Due Levis Step Levis Le	
Dec per area zeros generos gen	
Dec Len Jeres Leverg Explose Enrore E	
Der Lein Stein Zeiten Z	
Der Lein Sing Levog Agnoss Ennow [nep] 7.Count no of nodes 8.Traverse reverse 10.Quit Enter data:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at beginning 3.Insert at beginning 5.Delete at beginning 5.Delete at beginning 5.Delete at end 4.Delete at designing 5.Delete at end 7.Count no of nodes 8.Traverse reverse 10.Quit Enter choice:8 35	
Der Lein Steig Leinig Agnors Zindow nep 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at position 3.Insert at beginning 5.Delete at beginning 5.Delete at beginning 5.Delete at nosition 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 35 25	
Dec Len Jene Levery Explose Endow Teep 7.Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter choice:2 Enter position:2 Node inserted 2 1. Insert at beginning 2. Insert at position 3. Insert at beginning 5. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 35 2 25 15	
Dec Los Step Level Agricol Minor Imp 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1. Insert at beginning 2. Insert at position 3. Insert at nod 4. Delete at position 6. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse forward 9. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 35 2 25 15 1. Insert at beginning	
Dec Lon Sing Levery Equivor Lindow (hep) 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at beginning 5.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 35 2 25 15 1.Insert at beginning 2.Insert at beginning 2.Insert at beginning 2.Insert at beginning 2.Insert at beginning 2.Insert at beginning 2.Insert at position	
Dec Len areg Leverg Equation Enrorm (Enrorm (Enrorm)) 7. Count no of nodes 8. Traverse reverse 10. Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1. Insert at beginning 2. Insert at med 4. Delete at beginning 5. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse forward 9. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 35 2 25 15 1. Insert at beginning 2. Insert at beginning 2. Insert at beginning 2. Insert at beginning 2. Insert at beginning 3. Insert at deginning 3. Insert at end	
Dec Len Step Levery Explose Endow here A. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1. Insert at beginning 2. Insert at beginning 5. Delete at position 6. Delete at beginning 5. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 35 2 25 15 1. Insert at beginning 2. Insert at beginning 3. Insert at end 4. Delete at beginning 2. Insert at position 3. Insert at end 4. Delete at beginning 5. Delete at end 5. Delete at beginning 5. Delete at end 5. Delete at beginning 5. Delete at end 5. Delete at beginning 5. Delete at end 5. Delete at beginning 5. Delete at beginning 5. Delete at beginning 5. Delete at end 5. Delete at beginning 5. Delete at beginning	
Dec Len area Levery Equation Enrorm (Enrorm (Enrorm)) 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice: 2 Enter data.2 Enter position: 2 Node inserted 2 1. Insert at position 3. Insert at end 4. Delete at position 6. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse forward 9. Traverse forward 10. Quit Enter choice: 8 35 2 2 5 15 1. Insert at beginning 2. Insert at position 3. Insert at end 4. Delete at beginning 5. Delete at end 4. Delete at beginning 5. Delete at position 5. Delete at end 5.	
Dec Lon Sing Levery Equivor Enror (1997) C. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter data:2 Enter osition:2 Node inserted 2 1. Insert at beginning 2. Insert at ned 4. Delete at beginning 5. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 35 2 25 15 1. Insert at beginning 2. Insert at beginning 2. Insert at position 3. Insert at beginning 2. Insert at position 3. Insert at beginning 2. Insert at position 3. Insert at objetion 3. Insert at objetion 3. Insert at objetion 3. Insert at objetion 5. Delete at position 5. Delete at position 6. Delete at end 7. Count no of nodes	
Inc. London of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter position:2 Node inserted 2 1. Insert at position 3. Insert at end 4. Delete at beginning 5. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Taken at beginning 1. Insert at ted 1. Quit Enter choice:8 35 2 15 1. Insert at beginning 2. Insert at position 3. Insert at end 4. Delete at beginning 5. Delete at position 1. Insert at beginning 5. Delete at beginning 5. Delete at beginning 5. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 35 2 5 5 15 1. Insert at beginning 7. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse forward 9. Traverse forward 9. Traverse terves 1. Traverse forward 9.	
Inc. London of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter data:2 Enter choice:2 Enter data: 1. Insert at beginning 2. Insert at position 3. Insert at end 4. Delete at position 6. Delete at end 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:8 15 1. Insert at beginning 2. Insert at position 3. Insert at end 4. Delete at end 7. Count no of nodes 8. Traverse reverse 10. Quit 1. Insert serverse 10. Quit 1. Insert at beginning 2. Insert at end 4. Delete at end 7. Count no of nodes 8. Traverse reverse 1. Insert serverse 1. Insert ser	
Ine Lend Step Level Applied Endow Imp 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:2 Enter data:2 Enter choice:3 1.Insert at beginning 2.Insert at beginning 5.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 35 2 25 15 1.Insert at beginning 2.Insert at beginning 2.Insert at beginning 3.Insert at beginning 3.Insert at beginning 3.Insert at beginning 5.Delete at position 3.Insert at beginning 5.Delete at position 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse forward 9.Traverse reverse 10.Quit bit bit bit bit bit bit bit bit bit b	
Dec Lon Step Level general gen	
Date Levis Step Levis Levis Levis Levis Levis Date Levis Levis Levis Levis Levis Levis 1. Count no of nodes 8. Traverse reverse 10. Quit Enter choice:2 Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse reverse 10.Quit Enter choice:8 35 2 15 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice	Lin 111 Cet 7
Line and area general g	Lr: 111 Cct 7 R ^A ∧ IEI ⊄ // ENG 111/2013 I
Line and are general ge	ער גער גער גער גער גער גער גער גער גער ג
Image: Second Second Endow Emp 7. Count no of nodes 8. Traverse forward 9. Traverse reverse 10. Quit Enter choice:2 Enter data:2 Enter position:2 Node inserted 2 1.Insert at beginning 2.Insert at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9. Traverse reverse 10.Quit Enter choice:8 35 2 2.Insert at position 3.Insert at end 4.Delete at position 3.Insert at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice	لد: 111 Cot 7 برگ ^ ∎ ⊄* رز ENG 111 Jul 2019

		~	1911
(∰ Python 3.7.3 Shell*		٥	×
The fort such forder for the form			^
R Travese forward			
9.Traverse reverse			
10.Quit			
Enter choice:3			
Enter data-45			
1.Insert at beginning			
2.Insert at position			
3. Insert at end			
5.Delete at position			
6.Delete at end			
7. Count no of nodes			
0. I favelse follward O Traverse reverse			
10.Quit			
Enter choice:8			
35			
25			
15			
45 Journal of Residual Sciences			
2. Insert at oscition			
3.Insert at end			
4.Delete at beginning			
5.Delete at position			
o.Detete at end 7 Count no of nodes			
8. Traverse forward			
9.Traverse reverse			
10.Quit			
Line choice		Ln: 141	Col: 13
		0:20	=
	11 J	ul 2019	
🛃 *Python 3.7.3 Shell*			×
	-		
File Edit Shell Debug Options Window Help			•
File Edit Shell Debug Options Window Help 5.Delete at position 6 Delete at and			^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes	_		^
File Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward		0	^
File Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Ouit			^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Futer choice:4	_		^
File Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35	_		^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning	_		^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at end	_		^
Ele Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at beginning	_	D	^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at edd 4.Delete at beginning 5.Delete at position	_	D	^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 6.Delete	_		^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward	_	D	^
File Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at nodition 3.Delete at end 4.Delete at beginning 5.Delete at nodition 6.Delete at end 7.Count no of nodes 8.Traverse reverse 10.Quit File enter the set of the se	_	D	^
Ele Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at beginning 3.Insert at beginning 5.Delete at position 6.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit	_		^
File Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8	_	B	^
File Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 25	_	B	^
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at ned 4.Delete at tend 5.Delete at tend 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15	_		~
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at end 4.Delete at end 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit F.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 15 45 Lucent at beginning	_		~
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at no 5.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at no 5.Delete at no 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at beginning	_		~
Ele Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 2 5 15 5 15 15 15 15 15 15 15 15 15 15 15	_		^
Ele Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice: 4 Node deleted 35 1.Insert at beginning 2.Insert at position 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Filter choice: 8.Traverse forward 9.Traverse reverse 10.Quit Filter choice: 2 25 15 45 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 2.Insert at position			•
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at end 4.Delete at beginning 5.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at beginning 2.Insert at beginning 2.Insert at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at beginning 2.Insert at beginning 2.Delete at position 5.Delete at od			
Elle Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at end 4.Delete at end 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit File end 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 2.Insert at position 3.Insert at end 4.Delete at end 7.Count no of nodes			
Ele Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end			
Be Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at end 4.Delete at beginning 5.Delete at position 3.Insert at end 4.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at end 4.Delete at beginning 2.Insert at position 3.Insert at end 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at end 4.Delete at beginning 5.Delete at beginning 5.Delete at beginning <tr< td=""><td></td><td></td><td></td></tr<>			
Bit Edit Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse forward 9.Traverse reverse 10.Quit 11.Sert at beginning 11.Sert at beginning 11.Sert at beginning 12.Sert at beginning 13.Sert at end 14.Sert at beginning 13.Sert at end 14.Sert at beginning 13.Sert at en			
File jdit Shell Debug Options Window Help 5.Delete at position 6.Delete at et ad 7.Count no of nodes 8.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 3.Insert at position 3.Delete at position 6.Delete at ed 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 21.Insert at beginning 2.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 2.Insert at position 6.Delete at end 7.Count no of nodes		Ln: 167	^ 7 Coi:0
File jält Shell Debug Options Window Help 5.Delete at position 6.Delete at end 7.Comt no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at position 0.Delete at end 7.Comt no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 5.Delete at end 7.Comt no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 45 1.Insert at beginning 2.Insert at position 3.Insert at end 4.Delete at beginning 5.Delete at position 6.Delete at end 7.Comt no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice		Ln: 167 0-21	A 7 Col: 0
Be fait Shell Dabug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Travene forward 9.Travense reverse 10.Quit 1.Insert at beginning 2.Insert at nod 4.Delete at beginning 5.Delete at nod 7.Count no of nodes 8.Travense forward 9.Travense reverse 10.Quit Enter choice:8 2 2 5 1 5 1 5 1 5 1 5 5 5 1 5 5 5 5 5 5	^ IC (√ / € ENG 11)	Ln: 167 0-21 ul 2019	× × √ 7 Col: 0
Be jaft Shell Dabug Options Window Help 5.Delete at position 6.Delete at end 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:4 Node deleted 35 1.Insert at beginning 2.Insert at no 7.Count no of nodes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice:8 2 25 15 15 21.Insert at beginning 2.Insert at no 10.Quit Enter choice:8 2 25 15 26 15 27 15 28 29 15 45 1.Insert at beginning 2.Insert at no 2.Betet at position 6.Delete at end 7.Count no fondes 8.Traverse forward 9.Traverse reverse 10.Quit Enter choice!	^ ■ ⊄* (% ENG 11)	Ln: 167 0-21 ul 2019	↑ 7 Col: 0 ₹

- a. What is double linkedlist
- b. How to represent a node in double linkedlist
- c. Differentiate between single and double linkedlist

LAB ASSIGNMENT:

- a. Write a program to insert a node at first , last and at specified position of double linkedlist?
- b. Write a program to eliminate duplicates from double linkedlist?
- c. Write a program to delete a node from first, last and at specified position of double linkedlist?

POST-LAB VIVA QUESTIONS:

a. How to represent double linkedlist?

b. How will you traverse double linkedlist?c. List the advantages of double linked list over singlelist?

WEEK – 9 IMPLEMENTATION OF STACK USING LINKED LIST

OBJECTIVE:

Write a Python script to implement Stack using linked list.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

create()

Define a 'Node' structure with two members data and next. Define a Node pointer 'top' and set it to NULL. Implement the main method by displaying Menu with list of operations and make suitable function calls in the main method.

push(value) - Inserting an element into the Stack

Create a newNode with given value. Check whether stack is Empty (top == NULL) If it is Empty, then set newNode \rightarrow next = NULL. If it is Not Empty, then set newNode \rightarrow next = top. Finally, set top = newNode.

pop() - Deleting an Element from a Stack

Check whether stack is Empty (top == NULL). If it is Empty, then display "Stack is Empty!!! Deletion is not possible!!!" and terminate the function If it is Not Empty, then define a Node pointer 'temp' and set it to 'top'. Then set 'top = top \rightarrow next'. Finally, delete 'temp'. (free(temp)).

display() - Displaying stack of elements

Check whether stack is Empty (top == NULL). If it is Empty, then display 'Stack is Empty!!!' and terminate the function. If it is Not Empty, then define a Node pointer 'temp' and initialize with top. Display 'temp \rightarrow data --->' and move it to the next node. Repeat the same until temp reaches to the first node in the stack. (temp \rightarrow next !=NULL). Finally! Display 'temp \rightarrow data --->NULL'.

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.

b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

class StackNode:

Constructor to initialize a node
def_init_(self,data):
 self.data = data
 self.next =None

class Stack:

Constructor to initialize the root of linked list def_init_(self): self.root = None

```
def isEmpty(self):
return True if self.root is None else False
```

def push(self, data): newNode = StackNode(data) newNode.next = self.root self.root = newNode print ("%d pushed to stack" %(data))

def pop(self):
 if (self.isEmpty()):
 return float("-inf")
 temp = self.root
 self.root = self.root.next
 popped = temp.data
 return popped

def peek(self): if self.isEmpty(): returnfloat("-inf") returnself.root.data

Driver program to test above class
stack =Stack()
stack.push(10)
stack.push(20)
stack.push(30)

print ("%d popped from stack" %(stack.pop()))
print ("Top element is %d " %(stack.peek()))

Output:

Dutton 2.7.2 Shall	- a x
The file field Dates Output Wedge Units	
File Edit Shell Debug Options Window Help Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32 Type "help", "copyright", "credits" or "license()" for more information. >>> RESTART: C.'Users'Sirisha\AppData\Local\Programs\Python\Python37-32\Scripts\stack_LL.py 10 pushed to stack 20 pushed to stack 30 pushed to stack 30 popped from stack Top element is 20 >>>	
	Le: 10 Cel
	Ln: 10 Col: 10:17 —
	🕺 🔨 📼 🕸 🤅 ENG 👖 Jul 2019 📮

PRE-LAB VIVA QUESTIONS:

- a. What do you mean by stack overflow?
- b. What are the basic operations of astack?
- c. How to implementstack?

LAB ASSIGNMENT:

- a. Formulate a program to reverse a list of numbers usingstack.
- b. Write a program to find the factorial of a number usingstack.
- c. Develop a program to check a given expression is balanced or not usingstack

POST-LAB VIVA QUESTIONS:

- a. How to remove an element fromstack?
- b. How to insert an element using astack?
- c. Is it possible to store any number of data elements instack?
- d. What are the demerits ofstack?

WEEK – 10 IMPLEMENTATION OF QUEUE USING LINKED LIST

OBJECTIVE:

Write a Python program to implement Linear Queue using linked list.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Queue using linked list:

Create():

Define a 'Node' structure with two members data and next. Define two Node pointers 'front' and 'rear' and set both to NULL. Implement the main method by displaying Menu of list of operations and make suitable function calls in the main method to perform user selected operation.

enQueue(value) - Inserting an element into theQueue

Create a newNode with given value and set 'newNode \rightarrow next' to NULL. Check whether queue is Empty (rear == NULL) If it is Empty then, set front = newNode and rear = newNode. If it is Not Empty then, set rear \rightarrow next = newNode and rear = newNode.

deQueue() - Deleting an Element from Queue

Check whether queue is Empty (front == NULL). If it is Empty, then display "Queue is Empty!!! Deletion is not possible!!!" and terminate from the function If it is Not Empty then, define a Node pointer 'temp' and set it to 'front'. Then set 'front = front \rightarrow next' and delete 'temp' (free(temp)).

display() - Displaying the elements of Queue

Check whether queue is Empty (front ==NULL). If it is Empty then, display 'Queue is Empty!!!' and terminate the function. If it is Not Empty then, define a Node pointer 'temp' and initialize with front. Display 'temp \rightarrow data --->' and move it to the next node. Repeat the same until 'temp' reaches to 'rear' (temp \rightarrow next != NULL). Finally! Display 'temp \rightarrow data ---> NULL'.

PROCEDURE:

a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.b. Execute : Go to Run ->Run module(F5)

SOURCE CODE:

class Node: def_init_(self,data): self.data=data self.next=None

class Queue: definit(self): self.front=None self.ctr=0

```
self.rear=None
  def Enqueue(self,data):
    node=Node(data)
    if self.front==None:
      self.front=node
      self.rear=node
    else:
      self.rear.next=node
      self.rear=node
    print("Node enqueued to queue",data)
    self.ctr+=1
    return
  def Dequeue(self):
    if self.front==None:
      print("No Nodes exist")
    else:
      print("Dequeued from queue",self.front.data)
      self.front=self.front.next
      self.ctr-=1
    return
 def Traverse(self):
    if self.front==None:
     print("No Nodes exist")
     return
   temp=self.front
   while temp is not None:
     print(temp.data)
     temp=temp.next
def Menu():
  print("1.Enqueue\n2.Dequeue\n3.Traverse\n4.Number of nodes\n5.Exit")
  ch=int(input("Enter choice:"))
  return ch
s=Queue()
while True:
  ch=Menu()
  if ch==1:
    data=input("Enter data:")
    s.Enqueue(data)
  elif ch==2:
    s.Dequeue()
  elif ch==3:
    s.Traverse()
  elif ch==4:
    print("Number of nodes",s.ctr)
  else:
    print('Quit')
    break
```

Output:

Python 3.7.3 Shell 2 Edit Shell Debug Options Window Help	1000	٥	×
pe "help", "copyright", "credits" or "license()" for more information.			
>> ESTART: C:\Users\Sirisha\AppData\Local\Programs\Python\Python37-32\Scripts\queue_LL.py			
**********Queue************************			
inqueue			
Jequeue Traverse			
Number of nodes			
žxit			
ter choice:1			
de enqueted to quete 15			
2nqueue			
Jequeue			
Travelse Number of nodes			
Exit Exit			
ter choice:1			
ter data.25			
And enqueued to queue 25			
Dequene			
fraverse			
Vumber of nodes			
ter choice:3			
1			
inqueue			
Jequeue Traverse			
Number of nodes			
Êxit			- 1
ter choice		Ln: 20	Col: 1
	10	Ln: 5:	
	4G 11 Ju	l 2019	-
Python 3.7.3 Shell	1.1	đ	\times
i Edit Shelj Debug Options Window Help			
Exit			
ter choice:1			
Ade data.55			
Enqueue			
Jequeue			
iraverse Number of roder			
Kindes of houses			
tter choice:2			
queued from queue 15			
inqueue			
Traverse			
Number of nodes			
žait			- 1
ter choice:s			
Jequeue Francesco			
Inverse Number of nodes			
Exit			
ter choice:4			
mber of nodes 2			
Anyoeue Decuence			
Inverse			
Number of nodes			
Exit			
Exit ter choice:5 it			
Exit tter choice:5 it ≫			
Exit ter choice:5 it >		Ln: 6	7 Col:

PRE-LAB VIVA QUESTIONS:

- a. Which principle is followed inqueue?
- b. What are the applications of queue?

LAB ASSIGNMENT:

- a. Write a program to implement Queue operations using linkedlist.
- b. Formulate a program to implement circular queue operations usingarrays.
- c. Write a program to implement a priorityqueue?

POST-LAB VIVA QUESTIONS:

- a. What is the advantage of circular queue over linearqueue?
- b. Where priority queues areused?
- c. What is DEQUE?

WEEK – 11 IMPLEMENTATION OF QUEUE USING LINKED LIST

OBJECTIVE:

a. Write a Python script to implement depth firstsearch

b. Write a Python script to implement breadth firstsearch

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Depth first search

- 1. Define a Stack of size total number of vertices in thegraph.
- 2. Select any vertex as starting point for traversal. Visit that vertex and push it on to theStack.
- 3. Visit any one of the non-visited adjacent vertices of a vertex which is at the top of stack and push it on to the stack.
- 4. Repeat step 3 until there is no new vertex to be visited from the vertex which is at the top of the stack.
- 5. When there is nonewvertex to visit the nuse backtracking and poponevertex from the stack.
- 6. Repeat steps 3, 4 and 5 until stack becomesEmpty.
- 7. When stack becomes Empty, then produce final spanning tree by removing unused edges from the graph

Breadth first search

- 1. Define a Queue of size total number of vertices in thegraph.
- 2. Select any vertex as starting point for traversal. Visit that vertex and insert it into the Queue.
- 3. Visit all the non-visited adjacent vertices of the vertex which is at front of the Queue and insert them into theQueue.
- 4. When there is no new vertex to be visited from the vertex which is at front of the Queue then delete that vertex.
- 5. Repeat steps 3 and 4 until queue becomesempty.
- 6. When queue becomes empty, then produce final spanning tree by removing unused edges from the graph

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.
- b. Execute : Go to Run ->Run module(F5)

SOURCECODE:

Depth firstsearch

from collections import defaultdict class Graph:

Constructor
def_init_(self):

default dictionary to store graph
self.graph = defaultdict(list)

function to add an edge to graph
def addEdge(self,u,v):
 self.graph[u].append(v)

A function used by DFS def DFSUtil(self,v,visited):

Mark the current node as visited and print it visited[v]= True print (v),

Recur for all the vertices adjacent to this vertex
for i in self.graph[v]:
 if visited[i] == False:
 self.DFSUtil(i, visited)

The function to do DFS traversal. It uses # recursive DFSUtil() def DFS(self,v):

Mark all the vertices as not visited
visited = [False]*(len(self.graph))

Call the recursive helper function to print # DFS traversal self.DFSUtil(v,visited)

Driver code # Create a graph given in the above diagram g = Graph() g.addEdge(0, 1) g.addEdge(0, 2) g.addEdge(1, 2) g.addEdge(2, 0) g.addEdge(2, 3) g.addEdge(3,3)

print ("Following is DFS from (starting from vertex 2)") g.DFS(2)

Output:



```
from collections import defaultdict class Graph:
```

```
# Constructor
def_init_(self):
```

default dictionary to store graph
self.graph = defaultdict(list)

function to add an edge to graph
def addEdge(self,u,v):
 self.graph[u].append(v)

Function to print a BFS of graph
def BFS(self, s):

Mark all the vertices as not visited
visited = [False]*(len(self.graph))

```
# Create a queue for BFS
queue = []
```

Mark the source node as visited and enqueue it
queue.append(s)
visited[s] = True

while queue:

Dequeue a vertex from queue and print it s = queue.pop(0) print (s) # Get all adjacent vertices of the dequeued # vertex s. If a adjacent has not been visited, # then mark it visited and enqueueit for i in self.graph[s]: if visited[i] == False: queue.append(i) visited[i] = True

Driver code # Create a graph given in the above diagram g = Graph() g.addEdge(0,1) g.addEdge(0, 2) g.addEdge(1, 2) g.addEdge(2, 0) g.addEdge(2, 3) g.addEdge(3, 3)

print ("Following is Breadth First Traversal (starting from vertex 2)") g.BFS(2)

Output:



PRE-LAB VIVA QUESTIONS:

- a. What is graph?
- b. List various ways of representations of graph?
- c. How many graph traversal algorithms arethere?

LAB ASSIGNMENT:

a. Find DFS traversal of the followinggraph



b. Deduce the time complexity of DFSalgorithm

- POST-LAB VIVA QUESTIONS:a. What is the advantage of circular queue over linearqueue?b. Where priority queues areused?

 - c. What is DEQUE?

WEEK – 12 BASICS OF PYTHON

OBJECTIVE:

Write a Python script to perform the following:

- a. Create a binary searchtree.
- b. Traverse the above binary search tree recursively in pre-order, post-order and in-order.
- c. Count the number of nodes in the binary search tree.

RESOURCE:

Python 3.7.3

PROGRAM LOGIC:

Binary search tree:

Create():

If root == NULL

return NULL;

If number == root->data

- return root->data;
- If number < root->data return search(root->left)
- If number > root->data
 - return search(root->right)

Inorder(tree):

1. Traverse the left subtree, i.e., callInorder(left-subtree)

- 2. Visit theroot.
- 3. Traverse the right subtree, i.e., callInorder(right-subtree)

Preorder(tree):

1. Visit theroot.

- 2. Traverse the left subtree, i.e., call Preorder(left-subtree)
- 3. Traverse the right subtree, i.e., callPreorder(right-subtree)

Postorder(tree):

- 1. Traverse the left subtree, i.e., callPostorder(left-subtree)
- 2. Traverse the right subtree, i.e., callPostorder(right-subtree)
- 3. Visit theroot.

Number of nodes in BST:

CountNodes(node x) set n=1 //global variable If x=NULL return 0 If(x->left!=NULL) n=n+1 CountNode(x->left) If(x->right!=NULL) n=n+1 CountNode(x->right) return n

PROCEDURE:

- a. Create : Open a new file in Python shell, write a program and save the program with .pyextension.
- b. Execute : Go to Run -> Run module(F5)

SOURCE CODE:

```
Binary search tree:
class Node:
   def init (self,info): #constructor of class
      self.info = info #information for node
      self.left = None #left leef
      self.right = None #right leef
      self.level = None #level nonedefined
   def_str_(self):
      return str(self.info) #return as string
class searchtree:
   def_init_(self): #constructor of class
      self.root = None
   def create(self,val): #create binary search tree nodes
      if self.root == None:
        self.root = Node(val)
      else:
        current = self.root
        while 1:
          if val < current.info:
            if current.left:
              current = current.left
            else:
              current.left = Node(val)
              break:
          elif val > current.info:
            if current.right:
              current = current.right
            else:
              current.right = Node(val)
              break;
          else:
            break
   def bft(self): #Breadth-First Traversal
      self.root.level = 0
      queue = [self.root]
      out = []
      current_level = self.root.level
      while len(queue) > 0:
        current_node = queue.pop(0)
        if current_node.level >current_level:
          current_level +=1
          out.append("\n")
        out.append(str(current_node.info) + "")
        ifcurrent_node.left:
```

current_node.left.level = current_level + 1
queue.append(current_node.left)
if current_node.right:
 current_node.right.level = current_level +1
 queue.append(current_node.right)

result= "".join(out)
print (result)

def inorder(self,node): if node is not None: self.inorder(node.left) print (node.info) self.inorder(node.right)

def preorder(self,node): if node is not None:

print (node.info)
self.preorder(node.left)
self.preorder(node.right)

def postorder(self,node): if node is not None: self.postorder(node.left) self.postorder(node.right) print (node.info)

#Driver code tree =searchtree() arr = [8,3,1,6,4,7,10,14,13] for i in arr: tree.create(i) print ('Breadth-First Traversal') tree.bft() print ('Inorder Traversal') tree.inorder(tree.root) print ('Preorder Traversal') tree.preorder(tree.root) print ('Postorder Traversal') tree.postorder(tree.root)

Output:



Count the number of nodes in BST:

class BinaryTree:

```
def_init_(self, data):
  self.data = data
  self.left = None
  self.right = None
def insert_left(self, new_data):
  if self.left == None:
     self.left = BinaryTree(new_data)
  else:
     t = BinaryTree(new_data)
     t.left = self.left
     self.left = t
def insert_right(self, new_data):
  if self.right == None:
     self.right = BinaryTree(new_data)
  else:
     t = BinaryTree(new_data)
     t.right = self.right
     self.right = t
def get_left(self):
  return self.left
def get_right(self):
  returnself.right
def set_data(self, data):
  self.data = data
```

```
def get_data(self):
    return self.data

def size(my_tree):
    if not my_tree:
        return 0
    return 1 + size(my_tree.get_left()) + size(my_tree.get_right())
#Driver Code
```

a = BinaryTree(1)
a.insert_left(2)
a.insert_right(3)
print(size(a))

Output:



PRE-LAB VIVA QUESTIONS:

- a. Define tree traversal and mention types of traversal?
- b. Define atree?
- c. Define height of atree?
- d. Define depth of atree?
- e. Define degree of anode?
- f. Define Degree of atree?
- g. Define Terminal node or leafnode?
- h. Define Non-terminalnode?
- i. DefineSibling?
- j. Define BinaryTree?
- k. Write the properties of BinaryTree?
- 1. Find the minimum and maximum height of abinarytree?

LAB ASSIGNMENT:

- a. FormulateaprogramtocreateaBinaryTreeofintegers?
- b. Write a recursive program, for traversing a binary tree in preorder, inorder andpostorder?
- c. Compose a non-recursive program, for traversing a binary tree in preorder, inorder andpostorder?

d. Write a program to check balance property of atree?

POST-LAB VIVA QUESTIONS:

- a. Write the balance factor of a Binary Tree?
- b. What is a spanningTree?
- c. Define a Complete BinaryTree?
- d. List out the applications of BinaryTree?
- e. Write the two approaches for Binary TreeTraversal?f. Write the various operations performed in the binary searchtree?
- g. List out few of the Application of treedata-structure?
 h. Define pre-order traversal.
 i. Define post-ordertraversal.

- j. Define in-order traversal.