DATA STRUCTURES JOURNAL

DHRUVI NAIK
SYIT ROLL NO – 3027

MALAD KANDIVALI EDUCATION SOCIETY'S

NAGINDAS KHANDWALA COLLEGE OF COMMERCE, ARTS & MANAGEMENT STUDIES & SHANTABEN NAGINDAS KHANDWALA COLLEGE OF SCIENCE

MALAD [W], MUMBAI – 64

AUTONOMOUS INSTITUTION

(Affiliated To University Of Mumbai)

Reaccredited 'A' Grade by NAAC | ISO 9001:2015 Certified

CERTIFICATE

Name: Ms Dhruvi Naik

Roll No: 377 Programme: BSc IT Semester: III

This is certified to be a bonafide record of practical works done by the above student in the college laboratory for the course **Data Structures (Course Code: 2032UISPR)** for the partial fulfilment of Third Semester of BSc IT during the academic year 2020-21.

The journal work is the original study work that has been duly approved in the year 2020-21 by the undersigned.

External Examiner

Mr. Gangashankar Singh
(Subject-In-Charge)

Date of Examination: (College Stamp)

Class: S.Y. B.Sc. IT Sem- III **Roll No: _377_**

Subject: Data Structures

INDEX

Sr No	Date	Topic	Sign
1	04/09/2020	Implement the following for Array: a) Write a program to store the elements in 1-D array and provide an option to perform the operations like searching, sorting, merging, reversing the elements. b) Write a program to perform the Matrix addition, Multiplication and Transpose Operation.	
2	11/09/2020	Implement Linked List. Include options for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.	
3	18/09/2020	Implement the following for Stack: a) Perform Stack operations using Array implementation. b. b) Implement Tower of Hanoi. c) WAP to scan a polynomial using linked list and add two polynomials. d) WAP to calculate factorial and to compute the factors of a given no. (i) using recursion, (ii) using iteration	
4	25/09/2020	Perform Queues operations using Circular Array implementation.	
5	01/10/2020	Write a program to search an element from a list. Give user the option to perform Linear or Binary search.	
6	09/10/2020	WAP to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.	
7	16/10/2020	Implement the following for Hashing: a) Write a program to implement the collision technique. b) Write a program to implement the concept of linear probing.	
8	23/10/2020	Write a program for inorder, postorder and preorder traversal of tree.	

Practical – 01

AIM - Implement the following for Array:

a. Write a program to store the elements in 1-D array and provide an option to perform the operations like searching, sorting, merging, reversing the elements.

Arrays consist of fixed-size data records that allow each element to be efficiently located based on its index. Because arrays store information in adjoining blocks of memory they're considered *contiguous* data structures (as opposed to a *linked* data structure like a linked list.) Python includes several array-like data structures in its standard library that each have slightly different characteristics

- list – Mutable Dynamic Arrays
- Tuple – Immutable containers
- Bytes – Immutable arrays of single bytes … etc

- If we want to store arbitrary objects, with mixed data types, *list* or a *tuple* object can be used. We can try out the *array.array* when we've numeric data and tight packing along with performance is important.
- We can use the built-in *str* objects to represent textual data as Unicode characters.
- Immutable *bytes* type, or *byte array* can come in handy when we want to store contiguous block of bytes.

CODE

Prac1.py - C:/Users/Asus/Desktop/Dhruvi/DS/Prac1.py (3.8.5)

File Edit Format Run Options Window Help

```
arr1=[12,35,42,22,1,6,54]
arr2=['hello','world']
arr1.index(35)
print(arr1)
arr1.sort()
print(arr1)
arr1.extend(arr2)
print(arr1)
arr1.reverse()
print(arr1)
```

OUTPUT

Python 3.8.5 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:/Users/Asus/Desktop/Dhruvi/DS/Prac1.py ==============
[12, 35, 42, 22, 1, 6, 54]
[1, 6, 12, 22, 35, 42, 54]
[1, 6, 12, 22, 35, 42, 54, 'hello', 'world']
['world', 'hello', 54, 42, 35, 22, 12, 6, 1]
>>>
```

b. Write a program to perform the Matrix addition, Multiplication and Transpose operations

In python matrix can be implemented as 2D list or 2D Array. Forming matrix from latter, gives the additional functionalities for performing various operations in matrix. These operations and array are defines in module "numpy".
Operation on Matrix :
1. add() :- This function is used to perform element wise matrix addition.
2. subtract() :- This function is used to perform element wise matrix subtraction.
3. divide() :- This function is used to perform element wise matrix division.
4. multiply() :- This function is used to perform element wise matrix multiplication.
5. dot() :- This function is used to compute the matrix multiplication, rather than element wise multiplication.
6. sqrt() :- This function is used to compute the square root of each element of matrix.
7. sum(x,axis) :- This function is used to add all the elements in matrix. Optional "axis" argument computes the column sum if axis is 0 and row sum if axis is 1.
8. "T" :- This argument is used to transpose the specified matrix.

CODE

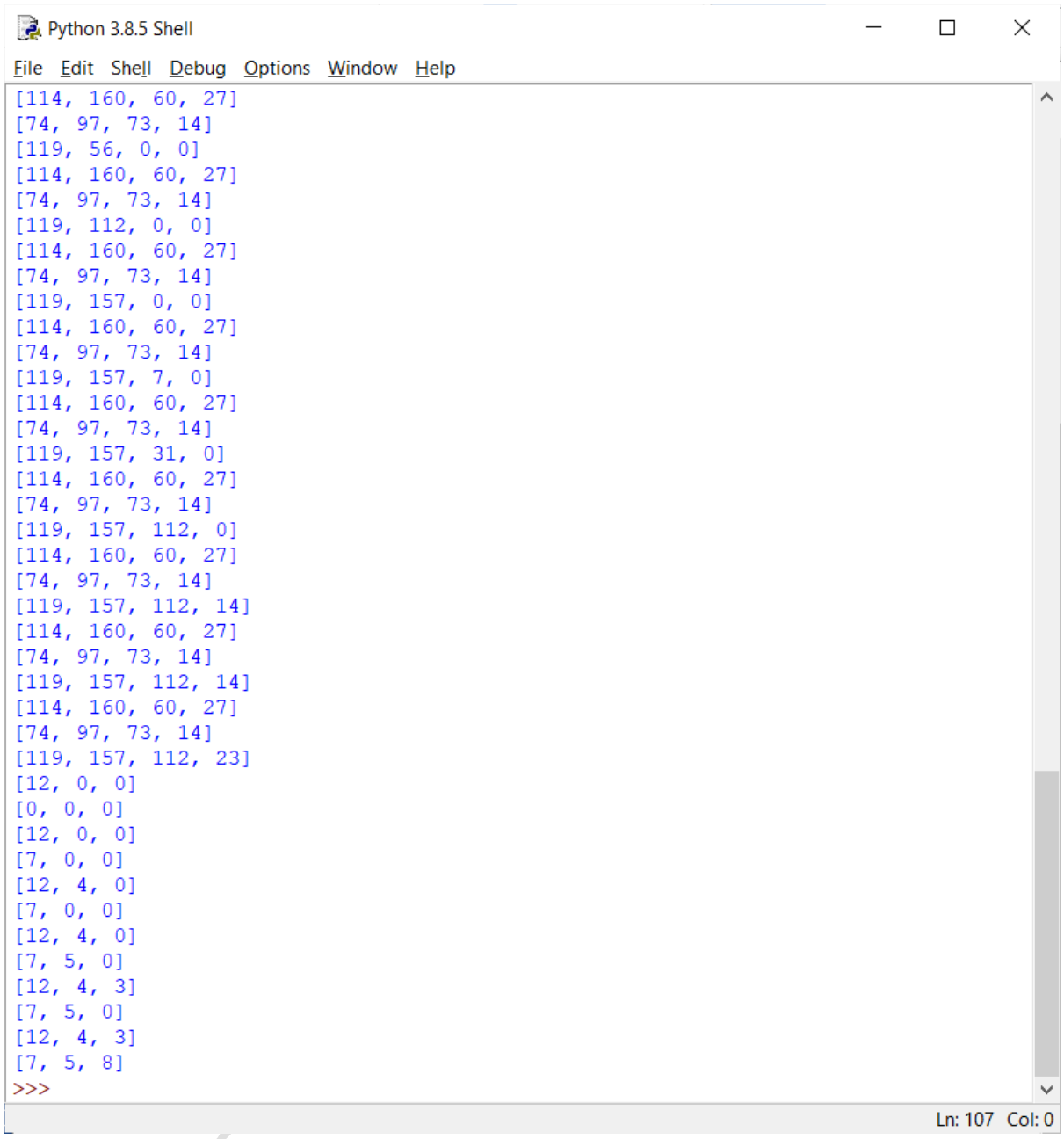
Prac1b.py - C:/Users/Asus/Desktop/Dhruvi/DS/Prac1b.py (3.8.5)

File Edit Format Run Options Window Help

```
# Program to add two matrices
X = [[11,7,3],
      [4 ,5,6],
      [7 ,8,9]]

Y = [[5,8,1],
      [6,7,3],
      [4,5,9]]

result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

# iterate through rows
for i in range(len(X)):
# iterate through columns
    for j in range(len(X[0])):
        result[i][j] = X[i][j] + Y[i][j]
p        for r in result:
            print(r)

# Program to multiply two matrices
 # 3x3 matrix
X = [[12,7,3],
     [4 ,5,6],
     [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
     [6,7,3,0],
     [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
          [0,0,0,0],
          [0,0,0,0]]
```

```
 # iterate through rows of X
for i in range(len(X)):
# iterate through columns of Y
    for j in range(len(Y[0])):
# iterate through rows of Y
        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]
            for r in result:
                print(r)

# Program to transpose a matrix
X = [[12,7], [4 ,5], [3 ,8]]
result = [[0,0,0], [0,0,0]]
 # iterate through rows
for i in range(len(X)):
# iterate through columns
    for j in range(len(X[0])):
        result[j][i] = X[i][j]
        for r in result:
            print(r)
```

OUTPUT


Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 56, 0, 0]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 112, 0, 0]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 0, 0]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 7, 0]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 31, 0]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 0]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 14]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 14]
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
[12, 0, 0]
[0, 0, 0]
[12, 0, 0]
[7, 0, 0]
[12, 4, 0]
[7, 0, 0]
[12, 4, 0]
[7, 5, 0]
[12, 4, 3]
[7, 5, 0]
[12, 4, 3]
[7, 5, 8]
>>>
Ln: 107 Col: 0

Practical – 2

AIM - 2. Implement Linked List. Include options for insertion, deletion and search of a number, reverse the list and concatenate two linked lists

A linked list is a sequence of data elements, which are connected together via links. Each data element contains a connection to another data element in form of a pointer. Python does not have linked lists in its standard library. We implement the concept of linked lists using the concept of nodes as discussed in the previous chapter. We have already seen how we create a node class and how to traverse the elements of a node.

Types of linked lists known as singly linked lists. In this type of data structure there is only one link between any two data elements. We create such a list and create additional methods to insert, update and remove elements from the list.

CODE

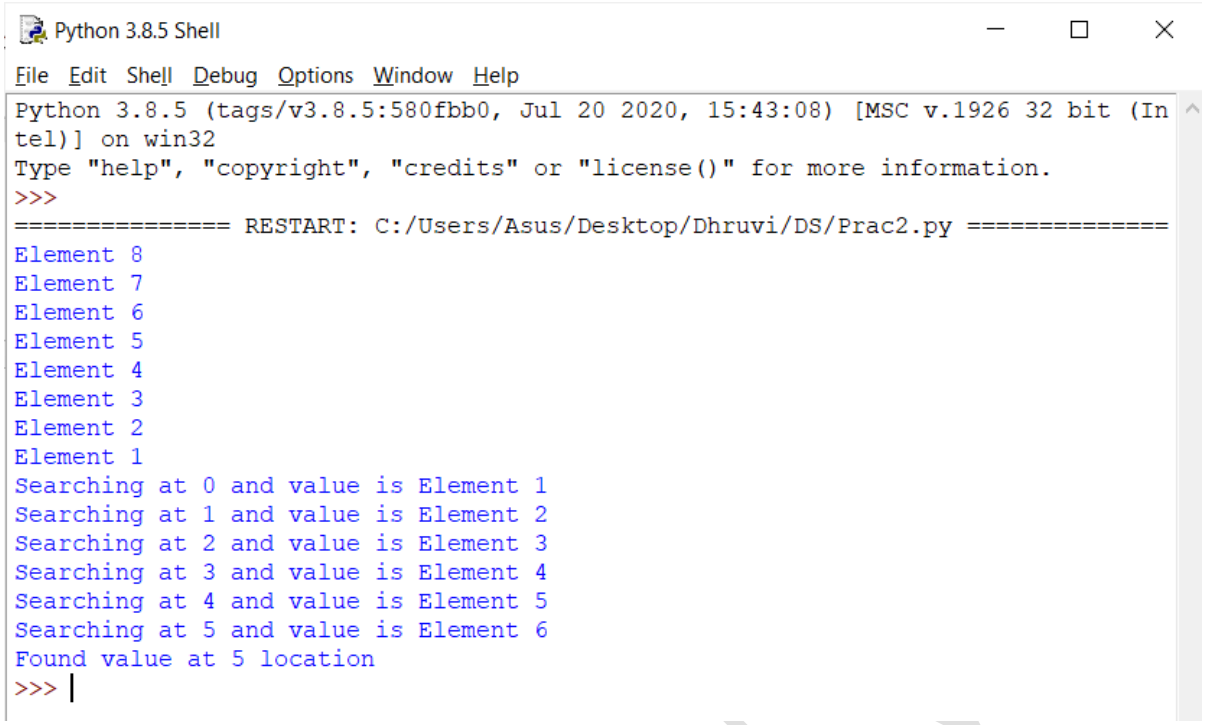
Prac2.py - C:/Users/Asus/Desktop/Dhruvi/DS/Prac2.py (3.8.5)

File Edit Format Run Options Window Help

```
class Node:

    def __init__ (self, element, next = None ):
        self.element = element
        self.next = next
        self.previous = None
    def display(self):
        print(self.element)

class LinkedList:

    def __init__(self):
        self.head = None
        self.size = 0



    def _len_(self):
        return self.size

    def get_head(self):
        return self.head


    def is_empty(self):
        return self.size == 0

    def display(self):
        if self.size == 0:
            print("No element")
            return
        first = self.head
        print(first.element.element)
        first = first.next
        while first:
            if type(first.element) == type(my_list.head.element):
                print(first.element.element)
                first = first.next
            print(first.element)
            first = first.next
```

Ln: 48 Col: 0

OUTPUT



Practical 3(a)

Aim: Implement the following for Stack:

a) Perform Stack operations using Array implementation.

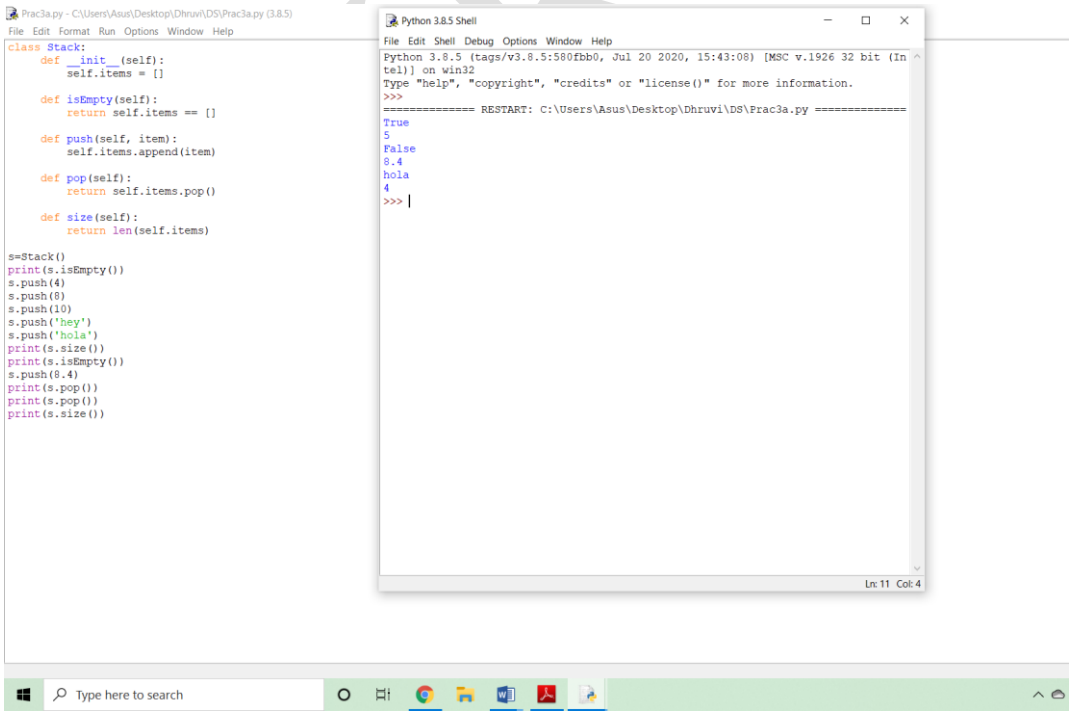
Theory:

Stacks is one of the earliest data structures defined in computer science. In simple words, Stack is a linear collection of items. It is a collection of objectsthat supports fast last-in, first-out (LIFO) semantics for insertion and deletion. It is an array or list structure of function calls and parameters used in modern computer programming and CPU architecture. Similar to a stack of plates at a restaurant, elements in a stack are added or removed from the top of the stack, in a “last in, first out” order. Unlike lists or arrays, random access is not allowed for the objects contained in the stack.

There are two types of operations in Stack:

- Push– To add data into the stack.
- Pop– To remove data from the stack

CODE AND OUTPUT



Practical 3(b)

Aim: Implement Tower of Hanoi.

Theory:

- We are given n disks and a series of rods, we need to transfer all the disks to the final rod under the given constraints
- We can move only one disk at a time.
- Only the uppermost disk.

CODE AND OUTPUT

Prac3b.py - C:/Users/Asus/Desktop/Dhruvi/DS/Prac3b.py (3.8.5)

File Edit Format Run Options Window Help

```
def Tower_of_Hanoi(disk , src, dest, auxiliary):
    if disk==1:
        print("Transfer disk 1 from source",src,"to destination",dest)
        return
    Tower_of_Hanoi(disk-1, src, auxiliary, dest)
    print("Transfer disk",disk,"from source",src,"to destination",dest)
    Tower_of_Hanoi(disk-1, auxiliary, dest, src)

disk = int(input("For how many rings you want to search.?"))
Tower_of_Hanoi(disk,'A','B','C')
```

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/Asus/Desktop/Dhruvi/DS/Prac3b.py ==============
For how many rings you want to search.?4
Transfer disk 1 from source A to destination C
Transfer disk 2 from source A to destination B
Transfer disk 1 from source C to destination B
Transfer disk 3 from source A to destination C
Transfer disk 1 from source B to destination A
Transfer disk 2 from source B to destination C
Transfer disk 1 from source A to destination C
Transfer disk 4 from source A to destination B
Transfer disk 1 from source C to destination B
Transfer disk 2 from source C to destination A
Transfer disk 1 from source B to destination A
Transfer disk 3 from source C to destination B
Transfer disk 1 from source A to destination C
Transfer disk 2 from source A to destination B
Transfer disk 1 from source C to destination B
>>>
```

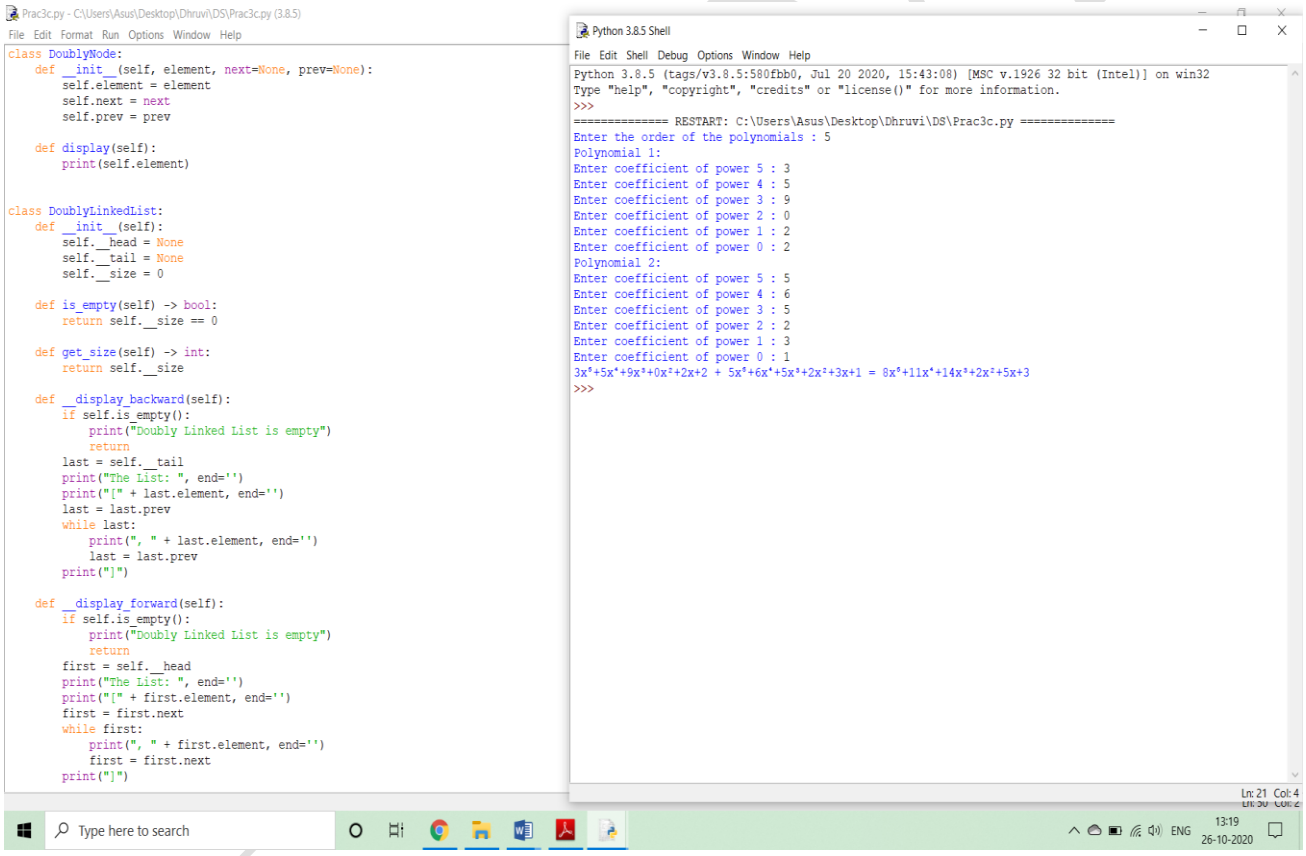
Practical 3(C)

Aim: WAP to scan a polynomial using linked list and add two polynomials.

Theory:

Polynomial is a mathematical expression that consists of variables and coefficients. for example x^2 - 4x +7. In the Polynomial linked list, the coefficients and exponents of the polynomial are defined as the data node of the list.For adding two polynomials that are stored as a linked list. We need to add the coefficients of variables with the same power. In a linked list node contains 3 members, coefficient value link to the next node a linked list that is used to store Polynomial looks like −Polynomial : 4x7 + 12x2 + 45

CODE AND OUTPUT



Practical 4

Aim: Perform Queues operations using Circular Array implementation.

Theory:

Circular queue avoids the wastage of space in a regular queue implementation using arrays. Circular Queue works by the process of circular increment i.e. when we try to increment the pointer and we reach the end of the queue, we start from the beginning of the queue. Here, the circular increment is performed by modulo division with the queue size. That is, if REAR + 1 == 5 (overflow!), REAR = (REAR + 1)%5 = 0 (start of queue) The circular queue work as follows:

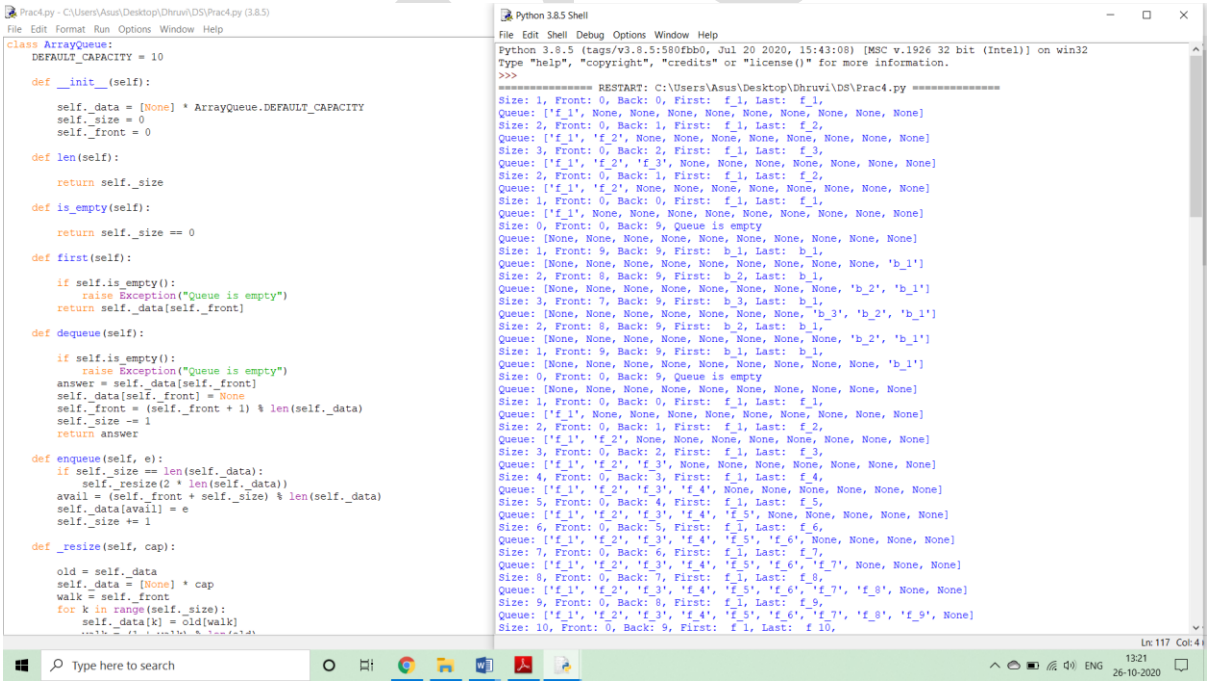
two pointers FRONT and REAR FRONT track the first element of the queue

REAR track the last elements of the queue initially, set value of FRONT and REARto -1

1. Enqueue Operation check if the queue is full for the first element, set value of FRONT to 0 circularly increase the REAR index by 1 (i.e. if the rear reaches the end, next it would be at the start of the queue) add the new element in the position pointed to by REAR

2. Dequeue Operation check if the queue is empty return the value pointed by FRONT circularly increase the FRONT index by 1 for the last element, reset the values of FRONT and REAR to -1

CODE AND OUTPUT

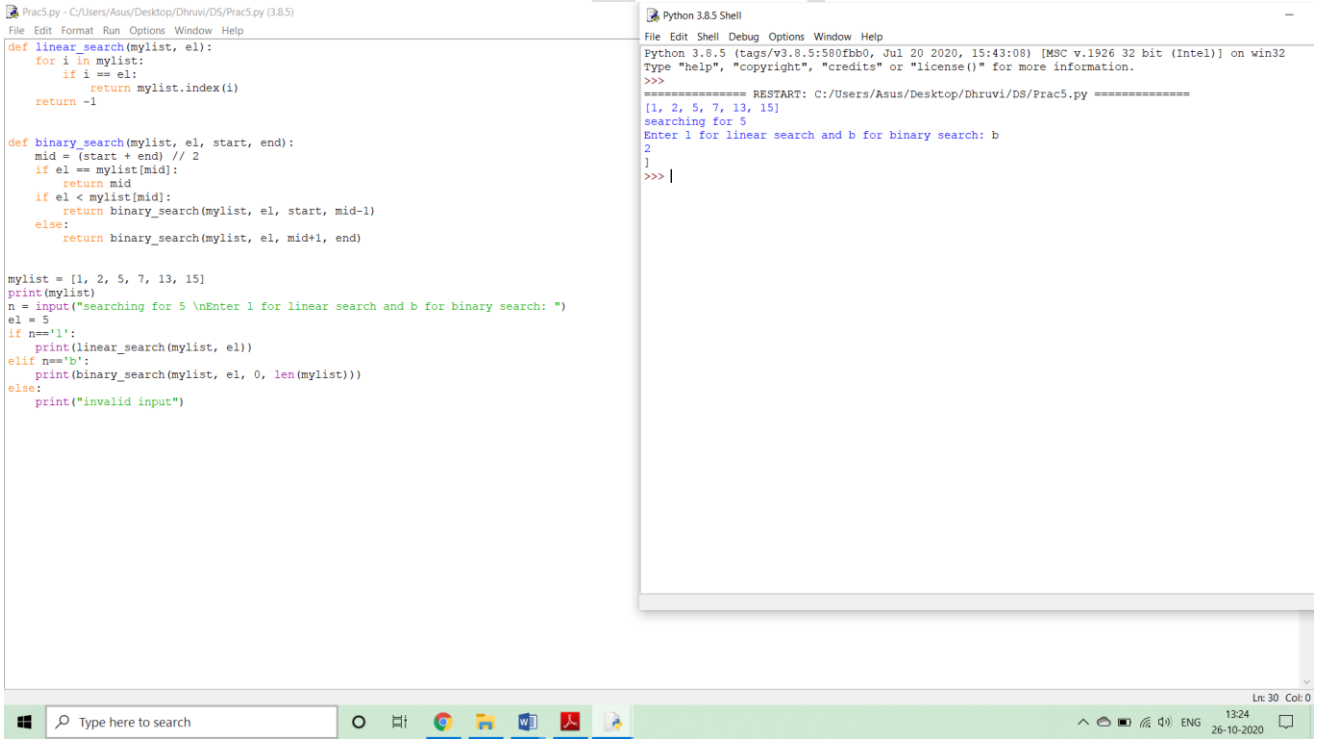


Aim: Write a program to search an element from a list. Give user the option to perform Linear or Binary search.

Theory:

- Linear Search: This linear search is a basic search algorithm which searches all the elements in the list and finds the required value. This is also known as sequential search.
- Binary Search: In computer science, a binary searcher half-interval search algorithm finds the position of a target value within a sorted array. The binary search algorithm can be classified as a dichotomies divide-and-conquer search algorithm and executes in logarithmic time.

CODE AND OUTPUT



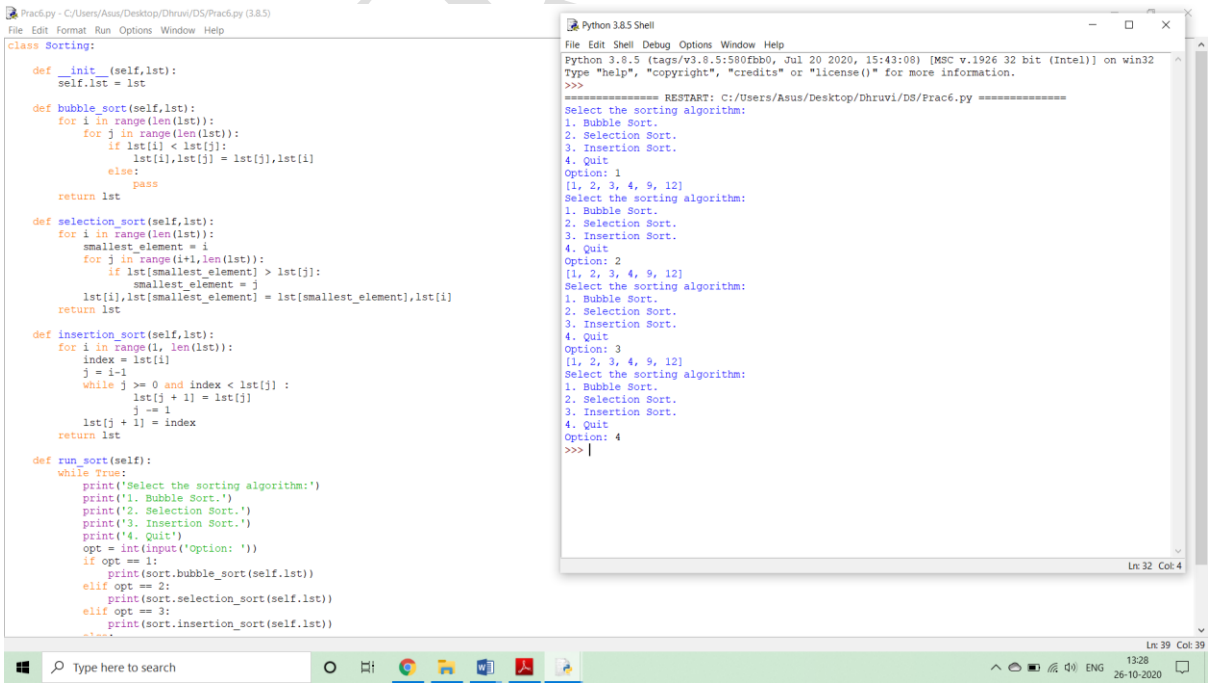
Practical 6

Aim: Write a program to search an element from a list. Give user the option to perform Linear or Binary search.

Theory:

- Bubble Sort: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.
- Selection Sort: The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two sub arrays in a given array
- Insertion Sort: Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

CODE AND OUTPUT



Practical 7(a)

Aim: Implement the following for Hashing:

Write a program to implement the collision technique.

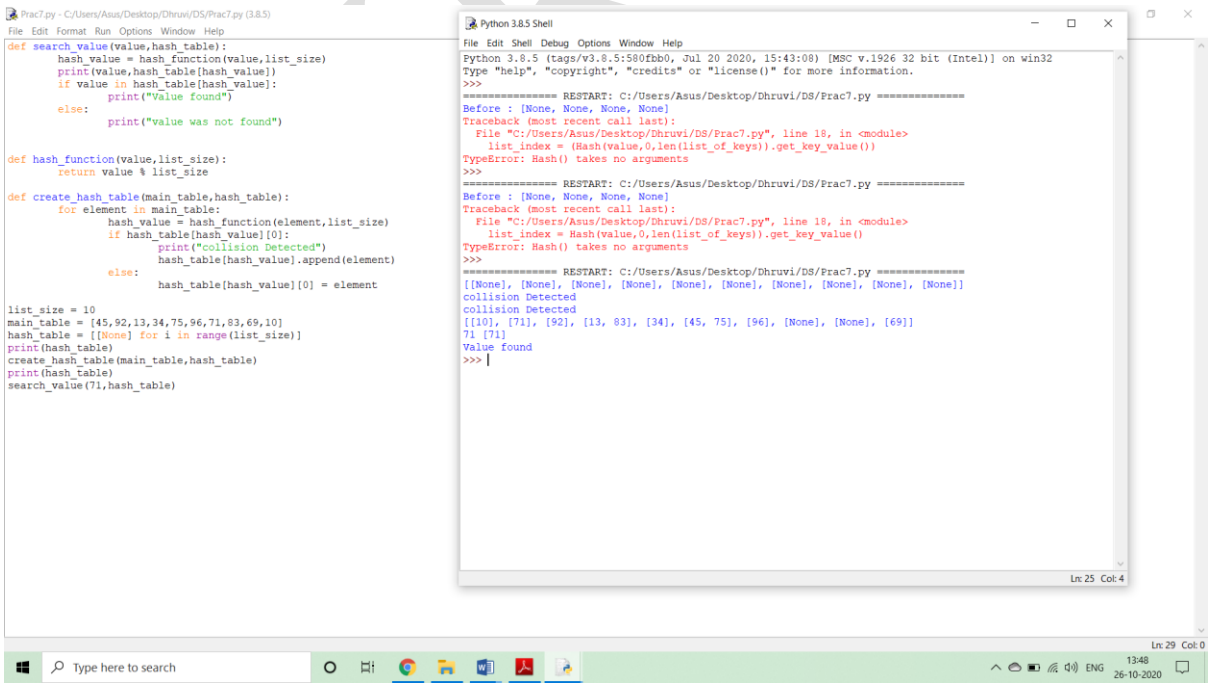
Theory:

Hashing:

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements. The efficiency of mapping depends of the efficiency of the hash function used.

- Collisions: A Hash Collision Attack is an attempt to find two input strings of a hash function that produce the same hash result. If two separate inputs produce the same hash output, it is called a collision.
- Separate Chaining: The idea is to make each cell of hash table point to a linked list of records that have same hash function value.
- Open Addressing: Like separate chaining, open addressing is a method for handling collisions. In Open Addressing, all elements are stored in the hash table itself. So at any point, the size of the table must be greater than or equal to the total number of keys (Note that we can increase table size by copying old data if needed)

CODE AND OUTPUT



Practical 7(b)

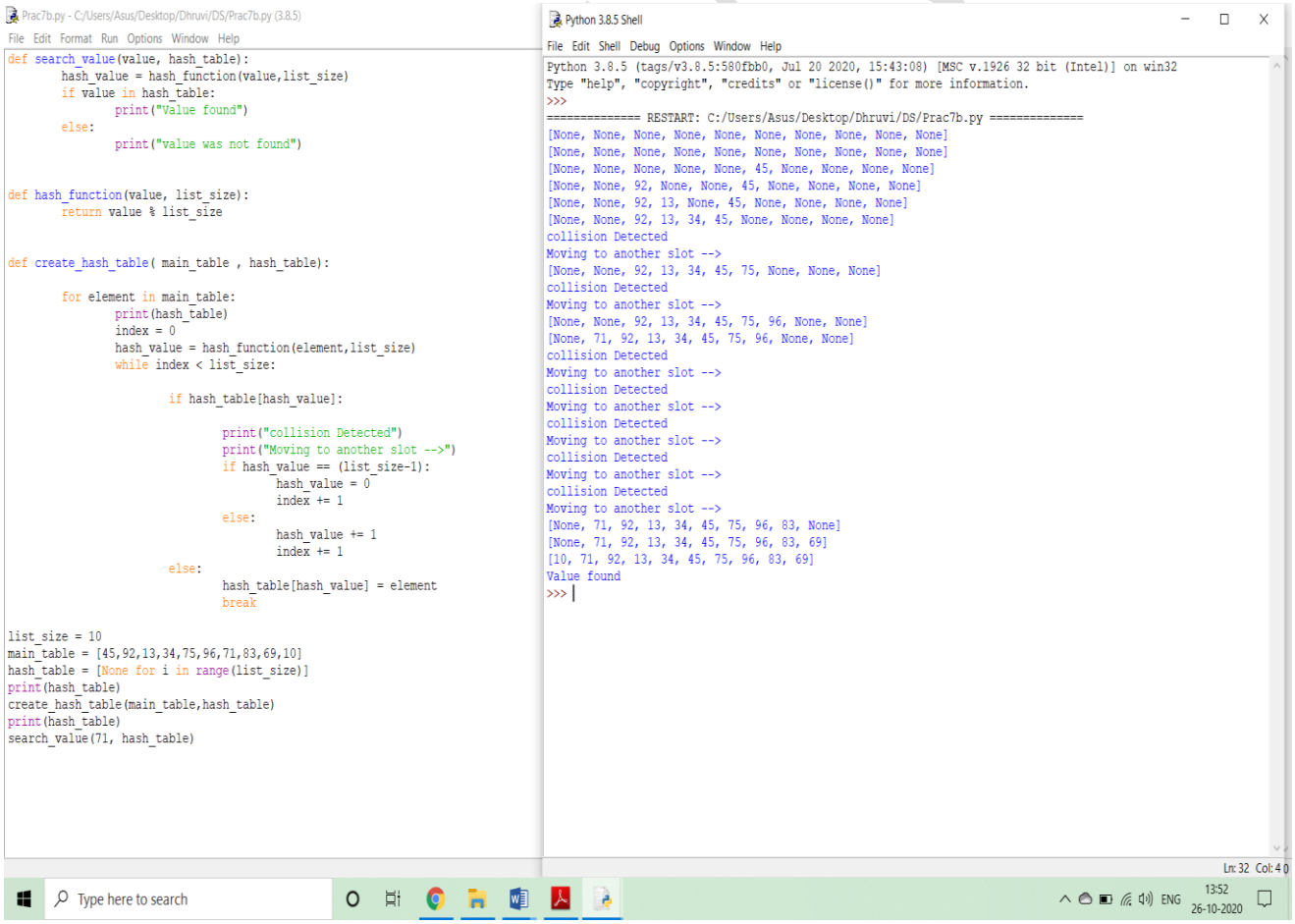
Aim: Implement the following for Hashing:

Write a program to implement the concept of linear probing.

Theory:

Linear probing is a scheme in computer programming for resolving collisions in hash tables, data structures for maintaining a collection of key–value pairs and looking up the value associated with a given key. Along with quadratic probing and double hashing,linear probing is a form of open addressing.

CODE AND OUTPUT



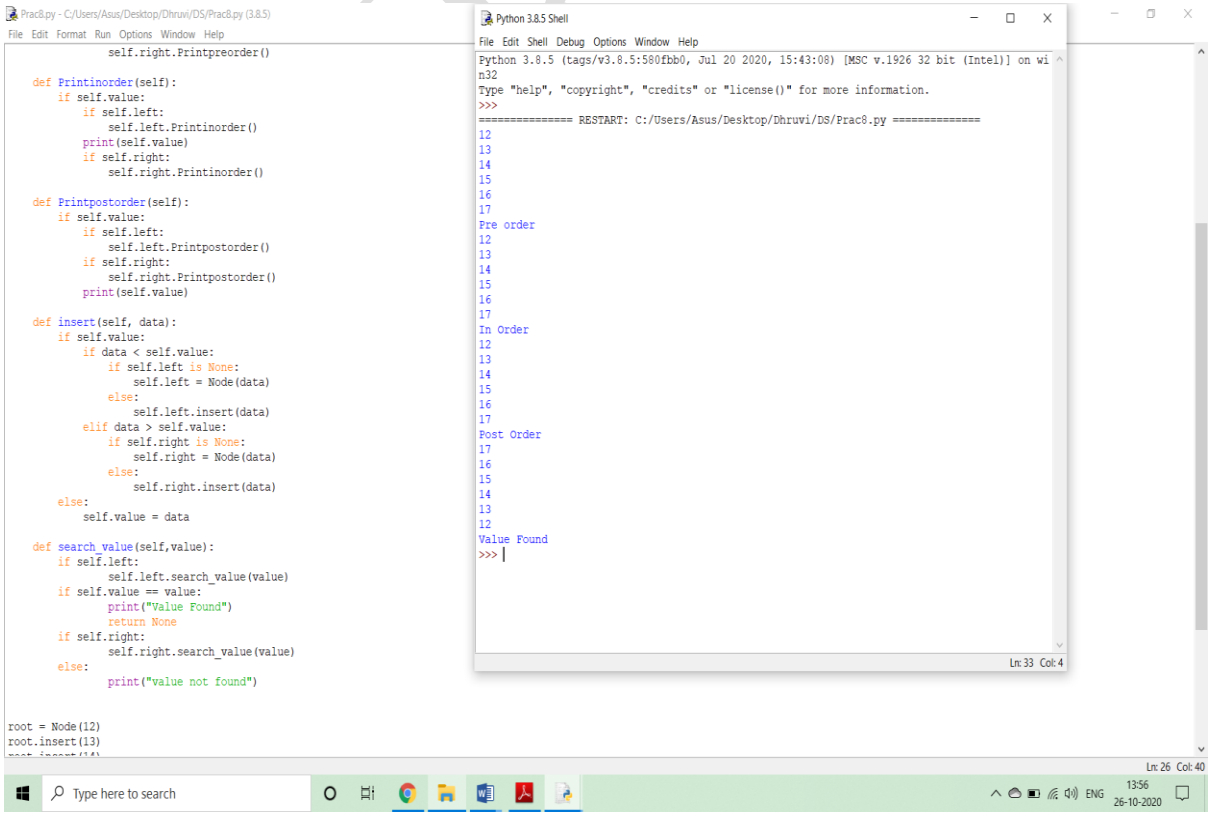
Practical 8

Aim: Write a program for inorder, postorder and preorder traversal of tree.

Theory:

- Inorder: In case of binary search trees (BST), Inorder traversal gives nodes in non-decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal s reversed can be used.
- Preorder: Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expression on of an expression tree.
- Postorder: Postorder traversal is also useful to get the postfix expression of an expression tree.

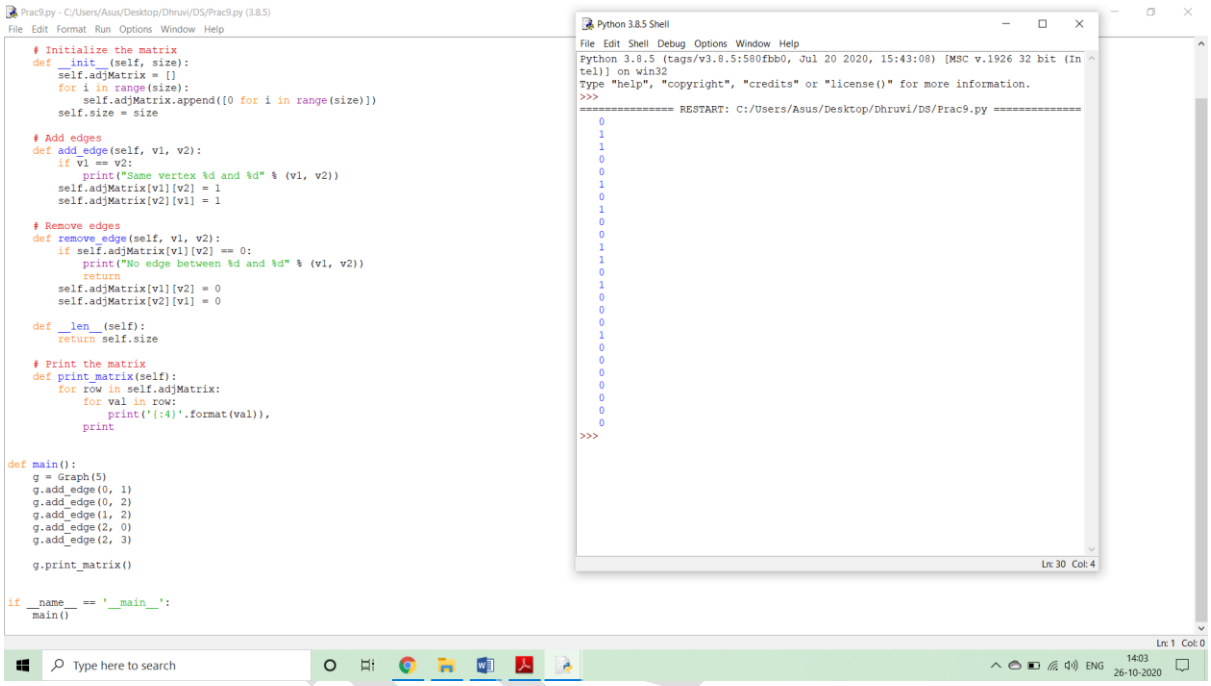
CODE AND OUTPUT



Practical – 9

AIM - Write a program to generate the adjacency matrix.

CODE AND OUTPUT



Practical – 10

AIM - Write a program for shortest path diagram.

CODE AND OUTPUT