

Data Structures File

Name: Ashish Gupta

Roll Number: 2K16/MC/023

Department of Applied Mathematics

Delhi Technological University

Subject Code: CS251

Content:

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Program Name** | **Teacher’s Signature** |
| **1** | Insertion and deletion at a particular index |  |
| **2** | Linear Search |  |
| **3** | Binary Search |  |
| **4** | Bubble Sort |  |
| **5** | Input elements in the array from C. Prompt and sort them |  |
| **6** | Stack operations-push/pop |  |
| **7** | Evaluation of postfix expression |  |
| **8** | Infix to postfix conversion |  |
| **9** | Infix to prefix conversion |  |
| **10** | Check for balanced parenthesis |  |
| **11** | String Operations- substring/concatenation/length |  |
| **12** | Enqueue an dequeue operations in queue |  |
| **13** | Implement insertion, deletion and traversal in circular queue. |  |
| **14** | Insertion in sorted linked list |  |
| **15** | Deletion from a linked list |  |
| **16** | Operations of doubly linked list |  |
| **17** | Stacks and Queues using linked list |  |
| **18** | Priority queue using linked list |  |
| **19** | Circular linked list & its operations |  |
| **20** | Creation of Binary Search Tree |  |
| **21** | Traversals in Binary Search Tree-Inorder/Preorder/PostOrder |  |
| **22** | Delete a node from BST |  |
| **23** | Traversal in a threaded binary tree |  |
| **24** | Breadth First Search |  |
| **25** | Depth First Search in a graph |  |
| **26** | Insertion Sort |  |
| **27** | Quick Sort |  |
| **28** | Merge Sort |  |
| **29** | Heap Sort |  |

Practical 1: Insertion and Deletion at particular index.

**LANGUAGE:** Python 3.7.2(a)

**CODE:**

myList=[]

num=int(input('Enter the number of elements in the array '))

print('Enter array')

for i in range(0,num):

myList.append(int(input('')))

print(myList)

Insert=int(input('Enter number to insert'))

Index=int(input('Enter index'))

if(Index>num):

print('Not in range')

elif Index<0:

print('Not a valid index')

else:

myList.insert(Insert,Index)

print(myList)

num=num+1

Index=int(input('Enter index at which you want to perform deletion '))

if(Index>=num):

print('Not in range')

elif Index<0:

print('Not a valid index')

else:

myList.remove(Index)

print(myList)

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

Enter the number of elements in the array 5

Enter array

1

2

3

4

5

[1, 2, 3, 4, 5]

Enter number to insert6

Enter index3

[1, 2, 3, 4, 5, 3]

Enter index at which you want to perform deletion 2

[1, 3, 4, 5, 3]

Practical 2: Linear Search.

**LANGUAGE:** Python 3.7.2(a)

**CODE:**

myList=[]

num=int(input('Enter the number of elements in the array: '))

print('Enter array')

for i in range(0,num):

myList.append(int(input('')))

print(myList)

search=int(input('Enter number to search : '))

flag=int(0)

for x in myList:

if x==search:

print('found')

flag=1

break

if(flag==0):

print('not found')

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

C:\Users\80LM0141IH\Desktop\Data structures Assignment\Linear search.py

Enter the number of elements in the array: 5

Enter array:

1

2

3

4

5

[1, 2, 3, 4, 5]

Enter number to search : 4

Found

Practical 3: Binary Search (Both Recursive and Iterative).

**LANGUAGE:** Python 3.7.2(a)

**CODE:**

def binarySearchRecursive(myList,element,start,end):

#base case

if(start>end): return False

mid=int((start+end)/2)

if myList[mid]==element:

return True

elif(myList[mid]>element):

return binarySearchRecursive(myList,element,start,mid-1)

else:

return binarySearchRecursive(myList, element, mid+1, end)

#the recursive case

def binarySearchIterative(myList,element,num):

start=0

end=num

while start<=end:

mid=int((start+end)/2)

if myList[mid] == element:

return True

elif (myList[mid] > element):

end=mid-1

else:

start=mid+1

return False

myList=[]

num=int(input('Enter the number of elements in the array '))

print('Enter array')

for i in range(0,num):

myList.append(int(input('')))

print(myList)

# my list has been printed here

myList.sort()

element=int(input('Enter Element to Search: '))

if(binarySearchRecursive(myList,element,int(0),int(num))==True):

print('Yes, It exists')

else:

print('No, It does not Exist')

if(binarySearchIterative(myList,element,num)==True):

print('Yes, It exists')

else:

print('No, It does not Exist')

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Binary Search(rec and iterative).py

Enter the number of elements in the array 4

Enter array

3

6

16

18

[3, 6, 16, 18]

Enter Element to Search: 4

No, It does not Exist

No, It does not Exist

Practical 4: Bubble Sort.

**LANGUAGE:** Python 3.7.2(a)

**CODE:**

def bubblesort(myList,num):

for i in range(0,num):

for j in range(i,num):

if(myList[i]>myList[j]):

temp=myList[i]

myList[i]=myList[j]

myList[j]=temp

return

myList=[]

num=int(input('Enter the number of elements in the array '))

print('Enter array')

for i in range(0,num):

myList.append(int(input('')))

print(myList)

bubblesort(myList,num)

print(myList)

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\BubbleSort.py

Enter the number of elements in the array 5

Enter array

5

4

3

100

2

[5, 4, 3, 100, 2]

[2, 3, 4, 5, 100]

Practical 5: Enter elements in array from command line and sort them.

**LANGUAGE:** C++(14)

**CODE:**

#include<iostream>

#include “bubblesort.h”

Using namespace std;  
int main(int argc, char \* argv[]){

string \*args = new string[argc];

for(int i = 0 ; i < argc ; i++)

args[i] = argv[i];

bubbleSort(args, argc);

for(int i = 0 ; i < argc ; i++)

cout << args[i] << endl;

return 0;

}

**Input and Output:**

Microsoft Windows [Version 10.0.15063]

(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\80LM0141IH\Desktop\DataStructuresFile\runCmdCode.c++> 6 5

4 3 1 5 6

1 3 4 5 6 /\* This is the output\*/

Practical 6: Stack Operations : Push and Pop.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class stack:

\_\_myList=[]

\_\_numElements=0

def \_\_init\_\_(self):

\_\_myList=[]

\_\_numElements=0

def Push(self, element):

self.\_\_myList.append(element)

self.\_\_numElements+=1

def Pop(self):

if(self.\_\_numElements>0):

self.\_\_myList.pop()

self\_\_numElements=self.\_\_numElements -1

def printStack(self):

print(self.\_\_myList)

def numElements(self):

return \_\_numElements

def get\_type(self):

print('stack')

# Driver program to test above function

s=stack()

s.Push(10)

s.printStack()

s.Push(20)

s.Push(30)

s.printStack()

s.Pop()

s.printStack()

s.Pop()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

= RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Stacks.py =

[10]

[10, 20, 30]

[10, 20]

[10]

>>>

Practical 7: Evaluation of Postfix Expression.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

# Python program to evaluate value of a postfix expression

# Class to convert the expression

class Evaluate:

# Constructor to initialize the class variables

def \_\_init\_\_(self, capacity):

self.top = -1

self.capacity = capacity

# This array is used a stack

self.array = []

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)))

return float(self.pop())

# Driver program to test above function

exp = "123/4\*+"

obj = Evaluate(len(exp))

print( "Value of %s is %d" % (exp, obj.evaluatePostfix(exp)))

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Evaluating PostFix.py

Value of 123/4\*+ is 3

>>>

Practical 8: Infix to Postfix Expression.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

from pythonds.basic.stack import Stack

def infixToPostfix(infixexpr):

prec = {}

prec["\*"] = 3

prec["/"] = 3

prec["+"] = 2

prec["-"] = 2

prec["("] = 1

opStack = Stack()

postfixList = []

tokenList = infixexpr.split()

for token in tokenList:

if token in "ABCDEFGHIJKLMNOPQRSTUVWXYZ" or token in "0123456789":

postfixList.append(token)

elif token == '(':

opStack.push(token)

elif token == ')':

topToken = opStack.pop()

while topToken != '(':

postfixList.append(topToken)

topToken = opStack.pop()

else:

while (not opStack.isEmpty()) and \

(prec[opStack.peek()] >= prec[token]):

postfixList.append(opStack.pop())

opStack.push(token)

while not opStack.isEmpty():

postfixList.append(opStack.pop())

return " ".join(postfixList)

print(infixToPostfix("A \* B + C \* D"))

print(infixToPostfix("( A + B ) \* C - ( D - E ) \* ( F + G )"))

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Evaluating PostFix.py >>> A B \* C D \* +  
A B + C \* D E - F G + \* -

Practical 9: Infix to Pretfix Expression.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

from pythonds.basic.stack import Stack

def infixToPretfix(infixexpr):

prec = {}

prec["\*"] = 3

prec["/"] = 3

prec["+"] = 2

prec["-"] = 2

prec["("] = 1

opStack = Stack()

postfixList = []

tokenList = infixexpr.split()

for token in tokenList:

if token in "ABCDEFGHIJKLMNOPQRSTUVWXYZ" or token in "0123456789":

postfixList.append(token)

elif token == '(':

opStack.push(token)

elif token == ')':

topToken = opStack.pop()

while topToken != '(':

postfixList.append(topToken)

topToken = opStack.pop()

else:

while (not opStack.isEmpty()) and \

(prec[opStack.peek()] >= prec[token]):

postfixList.append(opStack.pop())

opStack.push(token)

while not opStack.isEmpty():

postfixList.append(opStack.pop())

return reverse(" ".join(postfixList)) # **Here is the change from Postfix**

print(infixToPostfix(reverse("A \* B + C \* D")))#**Here is the change from Postfix Again**

print(infixToPostfix(reverse ("( A + B ) \* C - ( D - E ) \* ( F + G )")))#**Here too**

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Evaluating PREFIX.py >>>+ \* D C \* B A  
- \* + G F – E D \* C + B A

Practical 10: Check Balanced Parenthesis

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

import sys

import random

import os

def checkBalanced(myString):

myStack=list()

flag=0

for i in range(0 , len(myString)):

if(myString[i]=='{' or myString[i]=='(' or myString[i]=='<' or myString[i]=='[' ):

myStack.append(myString[i])

elif myString[i]=='}':

if len(myStack)!=0 and myStack[len(myStack)-1]=='{' and len(myStack)!=0:

myStack.pop()

continue

else :

return False

elif myString[i]=='>':

if len(myStack)!=0 and myStack[len(myStack)-1]=='<'and len(myStack)!=0:

myStack.pop()

continue

else :

return False

elif myString[i]==')':

if len(myStack)!=0 and myStack[len(myStack)-1]=='(' and len(myStack)!=0:

myStack.pop()

continue

else :

return False

elif myString[i]==']':

if len(myStack)!=0 and myStack[len(myStack)-1]=='[' and len(myStack)!=0:

myStack.pop()

continue

else :

return False

myString='a+(b+c)-d\*<e+f>'

if(checkBalanced(myString)==False):

print('Given string is :Not Balanced')

else :print('Given String is : Balanced')

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Balanced Parenthesis.py

Given string is : Balanced

Practical 11: String Operations

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

def mySubstring(string4,index1,index2):

for i in range(index1,index2+1):

print(string4[i],end="")

string1='Hello, I am Ashish.'

string2='I have not copied any practical in this file.'

string3='Hello Ashish, Great'

#Length of the string :

print('Length of the string 1 is : ', len(string1))

# lets concatinate

print('Concatinated string is : ')

string4=string1+string2

print(string4)

#substring

index1=10

index2=20

#lets find the substring in string 4

print('Substring is : ')

mySubstring(string4,index1,index2)

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\String Operations.py

Length of the string 1 is : 19

Concatinated string is :

Hello, I am Ashish.I have not copied any practical in this file.

Substring is :

m Ashish.I

Practical 12: Enqueue and Dequeue Operations in a Queue

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

import sys

import random

import os

class myQueue:

\_\_myList=[]

\_\_numElements=int(0)

def \_\_init\_\_(self):

\_\_myList=[]

\_\_numElements=int(0)

def enqueue(self,element):

self.\_\_myList.append(element)

self.\_\_numElements+=1

def dequeue(self):

if(self.\_\_numElements==0):

print('UnderFlow')

return

del(self.\_\_myList[0]) #check for errors

self.\_\_numElements -=1

def printQueue(self):

print(self.\_\_myList)

def countElements(self):

return self.\_\_numElements

queue=myQueue()

queue.enqueue(20)

queue.enqueue(40)

queue.enqueue(30)

queue.enqueue(10)

queue.printQueue()

queue.dequeue()

queue.dequeue()

queue.printQueue()

queue.enqueue(100)

queue.printQueue()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Queue Operations.py

[20, 40, 30, 10]

[30, 10]

[30, 10, 100]

>>>

Practical 13: Implement Insertion, Deletion and Traversing Circular Queue

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class myQueue:

\_\_myList=[None]\*5

\_\_front=int(0)

\_\_rear=int(0)

\_\_numElements=int(0)

\_\_sizeofQueue=int(0)

def \_\_init\_\_(self,Size):

self.\_\_\_\_myList=[None]\*Size

self.\_\_\_\_numElements=int(0)

self.\_\_\_\_sizeofQueue=Size

self.\_\_\_\_front = int(0)

self.\_\_\_\_rear = int(0)

def enqueue(self,element):

self.\_\_sizeofQueue=5

if(self.\_\_sizeofQueue==self.\_\_numElements):

print('OverFlow')

return

self.\_\_numElements+=1

self.\_\_myList[self.\_\_rear]=element

self.\_\_rear+=1

self.\_\_rear%=self.\_\_sizeofQueue

def dequeue(self):

if(self.\_\_numElements==0):

print('UnderFlow')

return

self.\_\_numElements-=1

self.\_\_front+=1

self.\_\_front%=self.\_\_sizeofQueue

def traverse(self):

if(self.\_\_front>=self.\_\_rear):

for i in range (self.\_\_front,self.\_\_sizeofQueue):

print(self.\_\_myList[i],end=" ")

for i in range(self.\_\_rear, self.\_\_front):

print(self.\_\_myList[i], end=" ")

else:

for i in range(self.\_\_front,self.\_\_rear):

print(self.\_\_myList[i],end=" ")

def countElements(self):

return self.\_\_numElements

**# This is the Driver Program that runs the code**

q=myQueue(5)

q.enqueue(10)

q.enqueue(90)

q.enqueue(60)

q.enqueue(30)

q.enqueue(40)

q.traverse()

q.enqueue(20)

q.dequeue()

q.dequeue()

q.dequeue()

q.enqueue(90)

q.enqueue(60)

q.enqueue(30)

q.traverse()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Circular queues.py

10 90 60 30 40

OverFlow

30 40

Practical 14: Insertion in a Sorted Linked List

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class Node:

def \_\_init\_\_(self,data=None,next=None):

self.data=data

self.next=next

def getData(self):

return self.data

def getNext(self):

return self.next

def setNext(self,newNext):

self.next=newNext

class sortedLL:

def \_\_init\_\_(self):

self.head=None

def Insert(self,data):

if(self.head==None):

newNode=Node(data)

self.head=newNode

return

prev=self.head

current=self.head

newNode=Node(data)

checker=int(data)

checker=int(data)

while current and current.getData()<=checker:

prev=current

current=current.getNext()

if current==None:

prev.setNext(newNode)

else:

prev.setNext(newNode)

newNode.setNext(current)

return

def Print(self):

current = self.head

while current:

print(current.getData(), end=" ")

current = current.getNext()

print('')

List=sortedLL()

List.Insert(10)

List.Print()

List.Insert(50)

List.Print()

List.Insert(30)

List.Print()

List.Insert(100)

List.Print()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Insertion in a Sorted Linked List.py

10

10 50

10 30 50

10 30 50 100

>>>

Practical 15: Deletion from a Linked List

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class Node:

def \_\_init\_\_(self,data=None,next=None):

self.data=data

self.next=next

def getData(self):

return self.data

def getNext(self):

return self.next

def setNext(self,newNext):

self.next=newNext

class LinkedList:

def \_\_init\_\_(self, head=None):

self.head = head

def Insert(self,data):

newNode=Node(data)

newNode.setNext(self.head)

self.head=newNode

def Size(self):

current=self.head

count=0

while current:

count=count+1

current=current.getNext()

return count

def Print(self):

current=self.head

while current:

print(current.getData(),end=" ")

current=current.getNext()

def search(self, data):

current = self.head

found = False

while current and found is False:

if current.getData() == data:

found = True

else:

current = current.getNext()

if current is None:

print('Not found %d in list' %(data))

return current

def delete(self, data):

current = self.head

previous = None

found = False

while current and found is False:

if current.getData() == data:

found = True

else:

previous = current

current = current.getNext()

if current is None:

print('Data Not in List')

return

if previous is None:

self.head = current.getNext()

else:

previous.setNext(current.getNext())

#**Driver Program for the code that is written**

List=LinkedList()

List.Insert(10)

List.Insert(50)

List.Insert(30)

List.Insert(100)

List.Insert(20)

List.Print()

List.delete(10)

List.delete(100)

print('')

List.Print()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Deletition from Linked List.py

20 100 30 50 10

20 30 50

>>>

Practical 16: Doubly Linked List and Its Operations

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class Node:

def \_\_init\_\_(self,data=None,next=None,prev=None):

self.data=data

self.next=next

self.prev=prev

def getData(self):

return self.data

def getNext(self):

return self.next

def setNext(self,newNext):

self.next=newNext

def setPrev(self,newPrev):

self.prev=newPrev

def getPrev(self):

return self.prev

class doublyLL:

def \_\_init\_\_(self, head=None):

self.head = head

def Insert(self,data): #the given program inserts at the front of the doubly linked list

if self.head==None:

newNode=Node(data)

self.head=newNode

return

newNode=Node(data)

newNode.setNext(self.head)

self.head.setPrev(newNode)

self.head=newNode

return

def Size(self):

current=self.head

count=0

while current:

count=count+1

current=current.getNext()

return count

def Print(self):

current=self.head

while current:

print(current.getData(),end=" ")

current=current.getNext()

def PrintReverse(self):

current=self.head

while current:

current=current.getNext()

while current:

print(current.getData(), end=" ")

current=current.getPrev()

print('')

return

def search(self, data):

current = self.head

found = False

while current and found is False:

if current.getData() == data:

found = True

else:

current = current.getNext()

if current is None:

print('Not found %d in list' %(data))

return current

def delete(self, data):

current = self.head

previous = None

found = False

while current and found is False:

if current.getData() == data:

found = True

else:

previous = current

current = current.getNext()

if current is None:

print('Data Not in List')

return

if previous is None:

self.head = current.getNext()

else:

previous.setNext(current.getNext())

current=current.setNext()

current.setPrev(previous)

#**Driver Program for the code that is written**

List=doublyLL()

List.Insert(100)

List.Insert(40)

List.Insert(30)

List.Insert(90)

List.Insert(25)

List.Insert(75)

List.Print()

List.PrintReverse()

List.delete(75)

List.delete(25)

print('List after deletion : ',end=" ")

List.Print()

print('')

print('Deleting 2 which is not in list : ',end=" ")

List.delete(2)#no such element in the doubly LL

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Doubly Linked Lists and Its Operations.py

75 25 90 30 40 100

List after deletion : 90 30 40 100

Deleting 2 which is not in list : Data Not in List

Practical 17: Stacks and Queues Using Linked List

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class Node:

def \_\_init\_\_(self,data=None,next=None,prev=None):

self.data=data

self.next=next

self.prev=prev

def getData(self):

return self.data

def getNext(self):

return self.next

def setNext(self,newNext):

self.next=newNext

def setPrev(self,newPrev):

self.prev=newPrev

def getPrev(self):

return self.prev

class stack:

def \_\_init\_\_(self):

self.size=0

self.head=None

def top(self):

if self.head==None:

print('Stack is empty.')

return

return self.head.getData()

def pop(self):

if self.head==None:

print('Stack is empty.')

return

def push(self,data):

newNode=Node(data)

if self.head==None:

self.head=newNode

return

newNode.setNext(self.head)

self.head=newNode

def print(self):

current=self.head

while current:

print(current.getData(),end=" " )

current=current.getNext()

print('')

def isEmpty(self):

if(head==None)

return True

return False

class Queue:

def \_\_init\_\_(self):

self.items = []

def isEmpty(self):

return self.items == []

def enqueue(self, item):

self.items.insert(0,item)

def dequeue(self):

return self.items.pop()

def size(self):

return len(self.items)

#**Driver Program for the code that is written**

myStack=stack()

myStack.push(10)

myStack.push(20)

myStack.push(100)

myStack.push(40)

myStack.push(30)

print('Top element of stack is : ',myStack.top())

myStack.pop()

print('printing whole stack : ')

myStack.print()

myStack.pop()

myStack.top()

if myStack.isEmpty()==True:

print('Stack is empty')

else:

print('Stack is non empty')

print('Start of Queues part')

q=Queue()

q.enqueue(4)

q.enqueue('dog')

q.enqueue(True)

print(q.size())

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Stacks and queues using LinkedLists.py

Top element of stack is : 30

printing whole stack :

30 40 100 20 10

Stack is non empty

Start of Queues part

3

[True, 'dog', 4] /\***This is a dynamic queue that stores variable object type variables in python** \*/

>>>

Practical 18: Priority Queues Using Linked List

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class Node( object ) :

def \_\_init\_\_( self, cargo = None, next = None ) :

self.cargo = cargo

self.next = next

class PriorityQueue :

#

def \_\_init\_\_( self ) :

self.length = 0

self.head = None

self.last = None

def isEmpty( self ) :

return (self.length == 0)

def enqueue( self, item, priority) :

newNode = Node(item)

newNode = Node(item)

newNode.next = None

if self.length == 0:

self.head =newNode

self.last = newNode

newNode.next = self.head

self.head = newNode

self.last.next = newNode

self.last = newNode

temp = self.head

p = self.head.next

while p != None :

if p.cargo > newNode.cargo:

temp = temp.next

p = p.next

break

newNode.next = temp.next

temp.next = newNode

def dequeue( self ) :

cargo = self.head.cargo

self.head = self.head.next

self.length = self.length - 1

if self.length == 0:

self.last = None

return cargo

def print(self):

current=self.head

while current:

print(current.cargo,end=" ")

current=current.next

print('')

return

#**Driver Program for the code that is written**

myQueue=PriorityQueue()

myQueue.enqueue(100,2)

myQueue.enqueue(130,1)

myQueue.enqueue(200,3)

myQueue.print()

myQueue.dequeue()

myQueue.print()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Priority queues using LinkedLists.py

130 100 200

>>>

Practical 19: Circular Linked List and Its Operations

**LANGUAGE:** C++(14)

**CODE:**

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node\*next;

///constructor

public:

Node()

{

next=NULL;

}

Node(int data)

{

this->data=data;

this->next=NULL;

}

};

class CircularLL

{

Node\*head;

/// Node\*tail;

int Size;

public:

///constructor

CircularLL()

{

head=NULL;

/// tail=NULL;

Size=0;

}

void Insert(int data,int pos)

{

Node\*newNode=new Node(data);

Size++;

if(head==NULL)

{

head=newNode;

}

Node\*temp;

while(temp->next!head)

{

temp=temp->next;

}

temp->next=newNode;

newNode->next=head;

head=newNode;

}

void Delete(int pos)

{

Node\*temp;

while(temp->next!head)

{

temp=temp->next;

}

temp->next=newNode;

newNode->next=head;

head=newNode;

}

}

bool isEmpty()

{

return Size==0;

}

int getSize()

{

return Size;

}

void print()

{

if(!head) return;

Node\*temp=head;

while(temp->next!=head)

{

cout<<temp->data<<" ";

temp=temp->next;

}

cout<<endl;

return;

}

};

#**Driver Program for the code that is written**

Int main()

{

CircularLL myLL;

myLL.Insert(10);

myLL.Insert(20);

myLL.Insert(100);

myLL.Insert(30);

myLL.Insert(60);

myLL.print();

myLL.Delete(2);

myLL.Delete(1);

myLL.print();

return 0;

}

**Input and Output:**

60 30 100 20 10

60 20 10

Practical 20: Creation of Binary Search Trees

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class treeNode:

data=0

left=None

right=None

def \_\_init\_\_(self,data):

self.data=data

self.left=None

self.right=None

def getData(self):

return self.data

def InsertHelper(root,data):

if(root==None):

newNode=treeNode(data)

return newNode

if data>=root.getData():

root.right= InsertHelper(root.right,data)

else:

root.left= InsertHelper(root.left,data)

return root

def PrintHelper(myRoot):

if myRoot == None:

return

print(myRoot.data)

PrintHelper(myRoot.left)

PrintHelper(myRoot.right)

class BST:

def \_\_init\_\_(self):

self.root=None

def PrintHelper(self,myRoot):

if myRoot == None:

return

print(myRoot.data)

PrintHelper(myRoot.left)

PrintHelper(myRoot.right)

def Insert(self, data):

self.root = InsertHelper(self.root, data)

def PrintTree(self):

PrintHelper(self.root)#check from here. I was last here

#**Driver Program for the code that is written**

myTree=BST()

myTree.Insert(10)

myTree.Insert(20)

myTree.Insert(5)

myTree.Insert(90)

myTree.PrintTree()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Binary Search Trees Creation.py

10

5

20

90

>>>

Practical 21: Traversal In A BST: Inorder Preorder and PostOrder Traversals

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class treeNode:

data=0

left=None

right=None

def \_\_init\_\_(self,data):

self.data=data

self.left=None

self.right=None

def getData(self):

return self.data

def getLeft(self):

return self.left

def getRight(self):

return self.right

def setLeft(self,newLeft):

self.left=newleft

def setRight(self,newRight):

self.right=newRight

def InsertHelper(root,data):

if(root==None):

newNode=treeNode(data)

return newNode

if data>=root.getData():

root.right=InsertHelper(root.right,data)

else:

root.left= InsertHelper(root.left,data)

return root

def PrintHelper(myRoot):

if myRoot == None:

return

print(myRoot.getData())

print(myRoot.data,end=" ")

PrintHelper(myRoot.left)

PrintHelper(myRoot.right)

def preOrder(myRoot):

if myRoot==None:

return

print(myRoot.data,end=" ")

inOrder(myRoot.left)

inOrder(myRoot.right)

def inOrder(myRoot):

if myRoot==None:

return

inOrder(myRoot.left)

print(myRoot.data,end=" ")

inOrder(myRoot.right)

def postOrder(myRoot):

if myRoot == None:

return

inOrder(myRoot.left)

inOrder(myRoot.right)

print(myRoot.data,end=" ")

class BST:

def \_\_init\_\_(self):

self.root=None

def Insert(self, data):

if self.root==None:

newNode=treeNode(data)

self.root=newNode

InsertHelper(self.root, data)

print(self.root, self.root.right)

def PrintTree(self):

PrintHelper(self.root)#check from here. I was last here

print('')

def PostOrderPrint(self):

postOrder(self.root)

print('')

def PreOrderPrint(self):

preOrder(self.root)

print('')

def InOrderPrint(self):

inOrder(self.root)

print('')

#**Driver Program for the code that is written**

myTree=BST()

myTree.Insert(10)

myTree.Insert(20)

myTree.Insert(5)

myTree.Insert(20)

myTree.Insert(100)

myTree.PrintTree()

myTree.InOrderPrint()

myTree.PreOrderPrint()

myTree.PostOrderPrint()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Binary Search Tree Travesals.py

10

10 5

5 10

10 20

20 20

20 100

100

5 10 10 20 20 100 # **These are the Traversals Inorder PreOrder and Postorder**

10 5 10 20 20 100

5 10 20 20 100 10

>>>

Practical 22: Delete a node from BST

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class treeNode:

data=0

left=None

right=None

def \_\_init\_\_(self,data):

self.data=data

self.left=None

self.right=None

def getData(self):

return self.data

def deleteHelper(root,data):

if root==None:

return

if root.getData()==data:

if root.left==None:

return root.right

elif root.right==None:

return root.left

else:

temp=root.left

while temp.right:

temp=temp.right

#swapping

x=root.data

root.data=temp.data

temp.data=x

root.left=deleteHelper(root.left,temp.data)

return root

else:

if root.getData()<data:

root.right= deleteHelper(root.right,data)

else:

root.left=deleteHelper(root.left,data)

def InsertHelper(root,data):

if(root==None):

newNode=treeNode(data)

return newNode

if data>=root.getData():

root.right= InsertHelper(root.right,data)

else:

root.left= InsertHelper(root.left,data)

return root

def PrintHelper(myRoot):

if myRoot == None:

return

print(myRoot.data,end=", ")

PrintHelper(myRoot.left)

PrintHelper(myRoot.right)

class BST:

def \_\_init\_\_(self):

self.root=None

def PrintHelper(self,myRoot):

if myRoot == None:

return

print(myRoot.data)

PrintHelper(myRoot.left)

PrintHelper(myRoot.right)

def Insert(self, data):

self.root = InsertHelper(self.root, data)

def PrintTree(self):

PrintHelper(self.root)#check from here. I was last here

print('')

def deleteElement(self,data):

if self.root==None:

print('UnderFlow')

return

self.root=deleteHelper(self.root,data)

# **This Is the driver program**

myTree=BST()

myTree.Insert(10)

myTree.Insert(20)

myTree.Insert(5)

myTree.Insert(90)

myTree.PrintTree()

myTree.deleteElement(10)

myTree.PrintTree()

myTree.PrintTree()

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Deletion in a BST.py

10, 5, 20, 90,

5, 20, 90,

5, 20, 90,

>>>

Practical 23: Traversal in a Threaded Binary Tree

**LANGUAGE:** C++ Code(Exclusive Code Blocks Compiler)

**CODE:**

void ThreadedBinaryTreeTraversal(struct node\* root)

{

    while (root)

    {

        // If left child is null, print the current node data. Move to

        // right child.

        if (root->left == NULL)

        {

            printf( "%d ", root->data );

            root = root->right;

        }

        else

        {

            // Find inorder predecessor

            struct node\* current = root->left;

            while (current->right && current->right != root)

                current = current->right;

            // If the right child of inorder predecessor already points to

            // this node

            if (current->right == root)

            {

                current->right = NULL;

                root = root->right;

            }

            // If right child doesn't point to this node, then print this

            // node and make right child point to this node

            else

            {

                printf("%d ", root->data);

                current->right = root;

                root = root->left;

            }

        }

    }

}

// Function for sStandard preorder traversal

void preorder(struct node\* root)

{

    if (root)

    {

        printf( "%d ", root->data);

        preorder(root->left);

        preorder(root->right);

    }

}

/\* Driver program to test above functions\*/

int main()

{

    struct node\* root = NULL;

    root = newNode(1);

    root->left = newNode(2);

    root->right = newNode(3);

    root->left->left = newNode(4);

    root->left->right = newNode(5);

    root->right->left = newNode(6);

    root->right->right = newNode(7);

    root->left->left->left = newNode(8);

    root->left->left->right = newNode(9);

    root->left->right->left = newNode(10);

    root->left->right->right = newNode(11);

    morrisTraversalPreorder(root);

    printf("\n");

    preorder(root);

    return 0;

}

**Input and Output:**

1 2 4 8 9 5 10 11 3 6 7

1 2 4 8 9 5 10 11 3 6 7

Process returned 0 (0x0) execution time : 0.076 s

Press any key to continue**.**

Practical 24 and 25: Breadth and Depth First Searches in a Graph.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

graph = {'A': set(['B', 'C']),

'B': set(['A', 'D', 'E']),

'C': set(['A', 'F']),

'D': set(['B']),

'E': set(['B', 'F']),

'F': set(['C', 'E'])

}

def dfs(graph, start):

visited, stack = set(), [start]

while stack:

vertex = stack.pop()

if vertex not in visited:

visited.add(vertex)

stack.extend(graph[vertex] - visited)

return visited

dfs(graph, 'A')

def dfs(graph, start, visited=None):

if visited is None:

visited = set()

visited.add(start)

for next in graph[start] - visited:

dfs(graph, next, visited)

return visited

dfs(graph, 'C') # {'E', 'D', 'F', 'A', 'C', 'B'}

def dfs\_paths(graph, start, goal):

stack = [(start, [start])]

while stack:

(vertex, path) = stack.pop()

for next in graph[vertex] - set(path):

if next == goal:

yield path + [next]

else:

stack.append((next, path + [next]))

list(dfs\_paths(graph, 'A', 'F')) # [['A', 'C', 'F'], ['A', 'B', 'E', 'F']]

def bfs(graph, start):

visited, queue = set(), [start]

while queue:

vertex = queue.pop(0)

if vertex not in visited:

visited.add(vertex)

queue.extend(graph[vertex] - visited)

return visited

bfs(graph, 'A') # {'B', 'C', 'A', 'F', 'D', 'E'}

def bfs\_paths(graph, start, goal):

queue = [(start, [start])]

while queue:

(vertex, path) = queue.pop(0)

for next in graph[vertex] - set(path):

if next == goal:

yield path + [next]

else:

queue.append((next, path + [next]))

list(bfs\_paths(graph, 'A', 'F')) # [['A', 'C', 'F'], ['A', 'B', 'E', 'F']]

#output graph = {'A': ['B', 'C'],'B': ['A', 'D', 'E'],'C': ['A', 'F'],'D': ['B'],'E': ['B', 'F'],'F': ['C', 'E']}

#output 2 = graph = {'A': ['B', 'C'],'B': ['A', 'D', 'E'],'C': ['A', 'F'],'D': ['B'],'E': ['B', 'F'],'F': ['C', 'E']}

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Insertion Sort.py

output = {'A': ['B', 'C'],'B': ['A', 'D', 'E'],'C': ['A', 'F'],'D': ['B'],'E': ['B', 'F'],'F': ['C', 'E']}

output 2 = graph = {'A': ['B', 'C'],'B': ['A', 'D', 'E'],'C': ['A', 'F'],'D': ['B'],'E': ['B', 'F'],'F': ['C', 'E']}

>>>

Practical 26:Insertion Sort.

**LANGUAGE:** Python(3.7.2(a))

**CODE: (Only Function)**

def insertionsort( aList ):

for i in range( 1, len( aList ) ):

tmp = aList[i]

k = i

while k > 0 and tmp < aList[k - 1]:

aList[k] = aList[k - 1]

k -= 1

aList[k] = tmp

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Depth First and Breadth First Search.py

#**The is just a Functions output that was run on Online IdeOne Editor…**

# Sample Input: [8,5,3,1,9,6,0,7,4,2,5]

# Sample Output: [0,1,2,3,4,5,5,6,7,8,9]

Practical 27: Quick Sort.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

def quickSort(alist):

quickSortHelper(alist,0,len(alist)-1)

def quickSortHelper(alist,first,last):

if first<last:

splitpoint = partition(alist,first,last)

quickSortHelper(alist,first,splitpoint-1)

quickSortHelper(alist,splitpoint+1,last)

def partition(alist,first,last):

pivotvalue = alist[first]

leftmark = first+1

rightmark = last

done = False

while not done:

while leftmark <= rightmark and alist[leftmark] <= pivotvalue:

leftmark = leftmark + 1

while alist[rightmark] >= pivotvalue and rightmark >= leftmark:

rightmark = rightmark -1

if rightmark < leftmark:

done = True

else:

temp = alist[leftmark]

alist[leftmark] = alist[rightmark]

alist[rightmark] = temp

temp = alist[first]

alist[first] = alist[rightmark]

alist[rightmark] = temp

return rightmark

#**This is the driver Function**

alist = [54,26,93,17,77,31,44,55,20]

quickSort(alist)

print(‘output: ‘,alist)

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\QuickSort.py

[14, 17, 13, 15, 19, 10, 3, 16, 9, 12]

output : [9, 3, 10, 13, 12, 14, 19, 16, 15, 17]

Practical 28: Merge Sort.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

def mergeSort(alist):

print("Splitting ",alist)

if len(alist)>1:

mid = len(alist)//2

lefthalf = alist[:mid]

righthalf = alist[mid:]

mergeSort(lefthalf)

mergeSort(righthalf)

i=0

j=0

k=0

while i < len(lefthalf) and j < len(righthalf):

if lefthalf[i] < righthalf[j]:

alist[k]=lefthalf[i]

i=i+1

else:

alist[k]=righthalf[j]

j=j+1

k=k+1

while i < len(lefthalf):

alist[k]=lefthalf[i]

i=i+1

k=k+1

while j < len(righthalf):

alist[k]=righthalf[j]

j=j+1

k=k+1

print("Merging ",alist)

#**This is the driver Function**

alist = [54,26,93,17,77,31,44,55,20]

mergeSort(alist)

print(‘Output: ‘,alist)

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\Merge Sort.py

Output: [17, 20, 26, 31, 44, 54, 55, 77, 93]

>>

Practical 29: Heap Sort.

**LANGUAGE:** Python(3.7.2(a))

**CODE:**

class Heap:

def \_\_init\_\_(self):

self.arr = []

self.Size = len(self.arr)

def maxHeapify(self, idx):

largest = idx

left = 2 \* idx + 1

right = 2 \* idx + 2

if left < self.Size and self.arr[left] > self.arr[largest]:

largest = left

if right < self.Size and self.arr[right] > self.arr[largest]:

largest = right

if not largest == idx:

self.arr[largest], self.arr[idx] = self.arr[idx], self.arr[largest]

self.maxHeapify(largest)

def buildHeap(self, array):

self.Size = len(array)

self.arr = array

i = (self.Size - 2) / 2

while i >= 0:

self.maxHeapify(i)

i -= 1

def heapSort(heap):

while heap.Size > 1:

heap.arr[0], heap.arr[heap.Size - 1] = heap.arr[heap.Size - 1], heap.arr[0]

heap.Size -= 1

heap.maxHeapify(0)

#**This is the driver Function**

myAns = map(int, input("Enter no's: ").split())

h = Heap()

h.buildHeap(myAns)

print(myAns)

heapSort(‘output : ‘ ,myAns)

print(myAns)

**Input and Output:**

Python 3.7.0a2 (v3.7.0a2:f7ac4fe, Oct 17 2017, 17:06:29) [MSC v.1900 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\80LM0141IH\Desktop\Data structures Assignment\heapsort.py

5

4

3

2

1

output: 1 2 3 4 5 6

>>>