**WEEK - 3**

**R-5.3**:-

import sys

data = []

cur = 0

for k in range(30):

a = len(data)

b = sys.getsizeof(data)

print('length:{0:3d};size in bytes:{1:4d}'.format(a, b))

data.append(None)

for k in range(30):

a = len(data)

b = sys.getsizeof(data)

print('length:{0:3d};size in bytes:{1:4d}'.format(a, b))

data.pop()

**R-5.7** :-

Sum of array - ((n-1)\*n)/2

**C-5.31** :-

def sumM(arr, x, n1, y, n2):

if(n1 == 1 and n2 ==1):

return arr[x][y]

if(n1 == 1):

return sumM(arr, x, n1, y, (n2/2))+sumM(arr, x, n1, y + (n2 / 2), n2 - (n2 / 2))

else:

return sumM(arr, x, (n1 / 2), y, n2) + sumM(arr, x + (n1 / 2), n1 - (n1 / 2), y, n2)

**R-6.1** :-

-> 5 — push(5)

-> 5 -> 3 — push(3)

-> 5 — pop()

->5 -> 2 — push(2)

->5 ->2 -> 8 — push(8)

->5 ->2 — pop()

->5 ->2 — pop()

->5 -> 2 ->9 — push(9)

->5 -> 2 ->9 ->1 — push(1)

->5 -> 2 ->9 — pop()

->5 -> 2 ->9 ->7 — push(7)

->5 -> 2 ->9 ->7 ->6 — push(6)

->5 -> 2 ->9 ->7 — pop()

->5 -> 2 ->9 — pop()

->5 -> 2 ->9 ->4 — push(4)

->5 -> 2 ->9 — pop()

->5 -> 2 — pop()

**R-6.3** :-

def change(arr1, arr2):

for i in range(len(arr1)):

arr2.append(arr1.pop())

print(arr2)

**R-6.5** :-

def reverseList(list):

stack = []

for i in list:

stack.append(i)

res =[]

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

res.append(stack.pop())

return res

**C-6.23** :-

import LimitedStack

Ls=LimitedStack.LimitedStack()

ls=LimitedStack.LimitedStack()

R = [1,2,3]#(or)[x for x in input("Enter an elements: ").split(',')]

S = [4, 5]#(or)[x for x in input("Enter an elements: ").split(',')]

T = [6, 7, 8, 9]#(or)[x for x in input("Enter an elements: ").split(',')]

for i in range(len(R)):

Ls.StackPush(R.pop())

R=Ls.Viewstack()

print('Status of Stack in R: ',R)

print("The top of the stack is : ",Ls.StackPeek())

for i in range(len(S)):

Ls.StackPush(S.pop())

R=Ls.Viewstack()

print('Status of Stack in R: ',R)

print("The top of the stack is : ",Ls.StackPeek())

for i in range(len(T)):

Ls.StackPush(T.pop())

R=Ls.Viewstack()

print('Status of Stack in R: ',R)

print("The top of the stack is : ",Ls.StackPeek())

for j in range(len(Ls.data)):

if (j > 2):

ls.StackPush(Ls.data.pop())

S=ls.Viewstack()

print('The Elements in stack S: ',S)

print('Top of a stack in S: ',ls.StackPeek())

print('Elements in Stack T: ',T)

**C-6.28** :-

class Empty(Exception):

pass

class ArrayStack:

"""LIFO Stack implementation using a Python list as underlying storage."""

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

"""Create an empty stack."""

self.\_data = [] # nonpublic list instance

self.maxlen=maxlen

def \_\_len\_\_(self):

"""Return the number of elements in the stack."""

return len(self.\_data)

def is\_empty(self):

"""Return True if the stack is empty."""

return len(self.\_data) == 0

def push(self, e):

"""Add element e to the top of the stack."""

if len(self.\_data) == self.maxlen:

raise Exception # Whatever exception you want

self.\_data.append(e) # new item stored at end of list

def top(self):

"""Return (but do not remove) the element at the top of the stack.

Raise Empty exception if the stack is empty.

"""

if self.is\_empty():

raise Empty('Stack is empty')

return self.\_data[-1] # the last item in the list

def pop(self):

"""Remove and return the element from the top of the stack (i.e., LIFO).

Raise Empty exception if the stack is empty.

"""

if self.is\_empty():

raise Empty('Stack is empty')

return self.\_data.pop()