Week-2

# Python program to Implement an ADT and compute space and time.

## Algorithm:

Step1: create class stack.

Step2: perform push operators using append function.

Step3: if stack is not empty perform deletion using pop method.

Step4: Display the elements of stack and find the size of stack using len() function. Step5: find the time complexity of the program.

## Program:

import time starts=time.time()

class Stack:

def \_\_init (self):

self.items = []

def isempty(self):

return self.items == []

def push(self, item): self.items.append(item)

print(item)

def pop(self):

return self.items.pop()

def peek(self):

return self.items[len(self.items) -1]

def size(self):

return len(self.items)

s=Stack()

print(s.isempty()) print("Push Operation") s.push(11)

s.push(12) s.push(13) time.sleep(2)

print("the topmost element is", s.peek()) print("Pop Operation")

print("The deleted element is",s.pop()) print("The deleted element is",s.pop()) print("size of stack is",s.size()) end=time.time()

print("run time of the program is :", end-start)

## Output:

True

Push Operation 11

12

13

the topmost element is 13 Pop Operation

The deleted element is 13 The deleted element is 12 size of stack is 1

run time of the program is :

# Python program to implement above solution using array and compute space and time complexities and compare two solutions.

## Algorithm:

Step1: create class stack.

Step2: perform push operators using insert function.

Step3: if stack is not empty perform deletion using pop method.

Step4: Display the elements of stack and find the size of stack using len() function. Step5: find the space and time complexity of the program.

## Program:

import time

start =time.time()

a=[1,2,3,4]

a.insert(4,5) print(a)

a.pop(0) print(a)

len=len(a)

print("length of the list:",len)

end=time.time()

print("runtime of the program is: ",end-start)

**Output:**

[1, 2, 3, 4, 5]

[2, 3, 4, 5]

length of the list: 4

runtime of the program is: 0.021986007690429688