PROGRAM-1

Python Program To Use And Demonstrate Basic Data Structures.

```
print("Demonstration of BASIC Data Structures")
l1=[159,26,"DS With Python",48623,"Lists in Python",578,6]
print("List is:")
print(l1)

d1={"OOPD with Java":"20CS42P","Operating System":"20CS43P","Software
Engineering":"20CS42P"}
print("Dictionary Contents are")
print(d1)

t1=(100,250,380,450,578,625)
print("Tuples contents are")
print(t1)

s1={180,620,740,460,534,357}
print("Set contents are")
print(s1)
```

OUTPUT

```
Demonstration of BASIC Data Structures
List is:
[159, 26, 'DS With Python', 48623, 'Lists in Python', 578, 6]
Dictionary Contents are
{'OOPD with Java': '20CS42P', 'Operating System': '20CS43P', 'Software Engineering':
'20CS42P'}
Tuples contents are
(100, 250, 380, 450, 578, 625)
Set contents are
{740, 357, 620, 460, 180, 534}
```

PROGRAM-2

Implement an ADT with all its operations.

Implementation of stack ADT

```
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]

s=Stack()
print(s.isEmpty())
print("Push operation")
s.push(11)
s.push(12)
s.push(13)
print("Peek element of stack is :",s.peek())
print("Pop operation")
print(s.pop())
```

OUTPUT

```
True
Push operation
11
12
13
Peek element of stack is: 13
Pop operation
13
```

Implementation of Queue ADT

```
class Queue:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def enqueue(self, item):
        self.items.append(item)
        print(item)

    def front(self):
        return self.items[len(self.items)-1]

    def dequeue(self):
        return self.items.pop(0)

q=Queue()
print(q.isEmpty())
print("Enqueue:Inserting Elements into Queue")
q.enqueue(11)
```

```
q.enqueue(12)
q.enqueue(13)
print("Front:Element at Front End",q.front())
print("Dequeue:Deleting Elements from Queue")
print(q.dequeue())
print(q.dequeue())
```

OUTPUT

```
True
Enqueue: Inserting Elements into Queue
11
12
13
Front:Element at Front End 13
Dequeue: Deleting Elements from Queue
11
12
```

PROGRAM-3

Implement an ADT and Compute space and time complexities.

```
class Student:
    def getStudentDetails(self):
        self.regno=input("Enter Register Number : ")
        self.name = input("Enter Name : ")
        self.DSWP =int(input("Enter DataStructure with Python Marks : "))
        self.OOPD= int(input("Enter Object Oriented Programming and Design with Java Marks : "))
        self.OSA= int(input("Enter Operating System and Administration Marks : "))
        self.SE= int(input("Enter Software Engineering Marks : "))

    def printResult(self):
        self.percentage = (float)( (self.DSWP + self.OOPD + self.OSA+self.SE) / 400 * 100 );
        print(self.regno, '\t',self.name,'\t', self.percentage)

import time
import sys
st=time.time()
S1=Student()
S1.getStudentDetails()
print("Result : ")
print("Reg no\t\t\t","Name\t","Percentage")
S1.printResult()
et=time.time()
elapsedtime=et-st
print("Time complexity of this code is:",elapsedtime)
spregno=sys.getsizeof(S1.regno)
spname=sys.getsizeof(S1.name)
spdswp=sys.getsizeof(S1.DSWP)
spoopd=sys.getsizeof(S1.OOPD)
sposa=sys.getsizeof(S1.OSA)
```

```
spse=sys.getsizeof(S1.SE)
sppercentage=sys.getsizeof(S1.percentage)
spacecomplexity=spregno+spname+spdswp+spoopd+sposa+spse
print("Space complexity of this code is",spacecomplexity)
```

OUTPUT

```
Enter Register Number : 159CS210058
Enter Name : VINUTHA G S
Enter DataStructure with Python Marks : 98
Enter Object Oriented Programming and Design with Java Marks : 96
Enter Operating System and Administration Marks : 95
Enter Software Engineering Marks : 94
Result :
Reg no          Name          Percentage
159CS210058     VINUTHA G S   95.75
Time complexity of this code is: 20.785562753677368
Space complexity of this code is 128
```

PROGRAM-4

Implement an ADT and Compute space and time complexities between Abstract Data Type and Array.

```
import time
from sys import getsizeof
import array as arr
start_time = time.time()
name=["Akash","Prakash","Manish"]
regno=arr.array('i',[1,2,3])
m1=arr.array('i',[20,25,30])
m2=arr.array('i',[25,20,35])
m3=arr.array('i',[30,35,40])
print("Student Details are")
for i in range(3):
    print("Name:",name[i])
    print("Register Number:",regno[i])
    print("Marks of student are",m1[i],m2[i],m3[i])
    average=(m1[i]+m2[i]+m3[i])/3
print("Average Marks of Student is",average)
print("space taken by program",
getsizeof(name)+getsizeof(regno)+getsizeof(m1)+getsizeof(m2)+getsizeof(m3))
print("time taken by program is",(time.time()-start_time))
```

OUTPUT

```
Student Details are
Name: Akash
Register Number: 1
Marks of student are 20 25 30
Name: Prakash
Register Number: 2
Marks of student are 25 20 35
```

Name: Manish

Register Number: 3

Marks of student are 30 35 40

Average Marks of Student is 35.0

Space taken by program 224

Time taken by program is 0.04686427116394043

PROGRAM-5

Implement Linear Search compute space and time complexities, plot graph using asymptomatic Notations.

```
import time
import sys
from sys import getsizeof
stime=time.time()
def linearSearch(array,n,x):
    for i in range(0,n):
        if(array[i]==x):
            return i
    return -1
array = [2,4,0,1,9]
x=9
n=len(array)
result=linearSearch(array, n, x)
if(result==-1):
    print("Element not found")
else:
    print("Element found at position: ",result+1)
print("Space taken by Program",getsizeof(array)+getsizeof(x)+getsizeof(n)+getsizeof(result))
print("Time taken by Program is",(time.time()-stime))
```

OUTPUT

Element found at position: 5

Space taken by program 98

Time taken by program is 0.00036454200744628906

PROGRAM-6

Implement Bubble, Selection, and Insertion sorting algorithms compute space and time complexities, plot graph using asymptomatic notations.

Bubble Sort

```
import time
import sys
from sys import getsizeof
stime=time.time()
def bubbleSort(arr):
    for i in range(len(arr)-1):
        for j in range(len(arr)-i-1):
            if arr[j] >arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    print(arr)
arrlist= [6,2,8,4,10,18,54,22,33,5,9,7,1,15]
print("Array before Sorting is")
print(arrlist)
print("Array after Sorting is")
bubbleSort(arrlist)
print("Space taken by Program",getsizeof(arrlist)+getsizeof(bubbleSort))
print("Time taken by Program is",(time.time()-stime))
```

OUTPUT

```
Array before Sorting is
[6, 2, 8, 4, 10, 18, 54, 22, 33, 5, 9, 7, 1, 15]
Array after Sorting is
[1, 2, 4, 5, 6, 7, 8, 9, 10, 15, 18, 22, 33, 54]
Space taken by Program 164
Time taken by Program is 0.000997304916381836
```

Selection Sort

```
import time
import sys
from sys import getsizeof
stime=time.time()
def selectionSort(arrlist):
    for i in range(len(arrlist)):
        min= i
        for j in range(i+1,len(arrlist)):
            if arrlist [min]>arrlist [j]:
                min=j
                arrlist[i],arrlist[min]=arrlist[min],arrlist [i]
    print(arrlist)
arrlist= [6,2,4,10,18,54,22,33,9,7,1,15]
print("Array before Sorting is")
print(arrlist)
print("Array after Sorting is")
selectionSort(arrlist)
print("Space taken by Program",getsizeof(arrlist)+getsizeof(selectionSort))
print("Time taken by Program is",(time.time()-stime))
```

OUTPUT

Array before Sorting is

[6, 2, 4, 10, 18, 54, 22, 33, 9, 7, 1, 15]

Array after Sorting is

[1, 4, 2, 6, 7, 9, 15, 10, 18, 33, 22, 54]

Space taken by Program 156

Time taken by Program is 0.015902042388916016

Insertion Sort

```
import time
import sys
from sys import getsizeof
stime=time.time()
def insertionSort(arrlist):
    for i in range(1,len(arrlist)):
        key=arrlist[i]
        j=i-1
        while j>=0 and key<arrlist[j]:
            arrlist[j+1]=arrlist[j]
            j=j-1
            arrlist[j+1]=key
    return arrlist
arrlist=[56,92,28,49,10,35,76,64]
print("Array before Sorting is ")
print(arrlist)
insertionSort(arrlist)
print("Array after Sorting is ")
print(arrlist)
print("Space taken by Program",getsizeof(arrlist)+getsizeof(insertionSort))
print("Time taken by Program is",(time.time()-stime))
```

OUTPUT

Array before Sorting is

[56, 92, 28, 49, 10, 35, 76, 64]

Array after Sorting is

[10, 28, 35, 49, 56, 64, 76, 92]

Space taken by Program 140

Time taken by Program is 0.015630006790161133

PROGRAM-7

Implement Binary Search using recursion Compute space and time complexities, plot graph using asymptomatic notations and compare two.

```
import time
import sys
from sys import getsizeof
stime=time.time()
def binary_search(arr,low,high,n):
    if low<=high:
        mid=(low+high)//2
        if arr[mid]==n:
            return mid
        elif arr[mid]>n:
            return binary_search(arr,low,mid-1,n)
        else:
            return binary_search(arr,mid+1,high,n)
    else:
        return -1
arr=[12,24,32,39,45,50,54]
n=45
res=binary_search(arr,0,len(arr)-1,n)
if res!=-1:
    print("Element is present at index",str(res))
else:
    print("Element is not present in array")
print("Space taken by Program",getsizeof(arr)+getsizeof(res))
print("Time taken by Program is",(time.time()-stime))
```

OUTPUT

```
Element is present at index 4
Space taken by Program 78
Time taken by Program is 0.0
```

PROGRAM-8

Implement singly linked list (Traversing the Nodes, searching for a Node, Prepending Nodes, and Removing Nodes)

```
class Node:
    def __init__(self, data):
        self.data=data
        self.next=None

class SinglyLinkedList:
    def __init__(self):
        self.head=None
        self.tail=None

    def addNode(self,data):
        if self.tail is None:
            self.head=Node(data)
            self.tail=self.head
        else:
            self.tail.next=Node(data)
            self.tail=self.tail.next

    def display(self):
        current=self.head
        while current is not None:
            print(current.data,end=' ')
            current=current.next

s=SinglyLinkedList()
n=int(input('Enter the number of elements in linked list : '))
for i in range(n):
    data=int(input('Enter data: '))
    s.addNode(data)
print('The linked list: ',end='')
```

```
s.display()
```

OUTPUT

```
Enter the number of elements in linked list : 5
Enter data: 11
Enter data: 12
Enter data: 13
Enter data: 14
Enter data: 15
The linked list: 11 12 13 14 15
```

Implement Fibonacci sequence with dynamic programing.

```
def fib(n):
    if n<=1:
        return n
    else:
        return fib(n-1) + fib(n-2)
n=int(input("Enter the term:"))
print("The Fibonacci value is:",fib(n))
```

OUTPUT

```
Enter the term:3
The Fibonacci value is: 2
```

PROGRAM-9

Python program to reverse a stack.

```
class Stack:
    def __init__(self):
        self.Elements =[]

    def push(self, value):
        self.Elements.append(value)

    def pop(self):
        return self.Elements.pop()

    def empty(self):
        return self.Elements == []

    def show(self):
        for value in reversed(self.Elements):
            print(value)

def BottomInsert(s, value):
    if s.empty():
        s.push(value)
    else:
        popped = s.pop()
        BottomInsert(s, value)
        s.push(popped)

def Reverse(s):
    if s.empty():
        pass
    else:
        popped = s.pop()
```

```
    Reverse(s)
    BottomInsert(s, popped)

stk = Stack()
stk.push(1)
stk.push(2)
stk.push(3)
stk.push(4)
stk.push(5)

print("Original Stack")
stk.show()

print("\nStack after Reversing")
Reverse(stk)
stk.show()
```

OUTPUT

```
Original Stack
5
4
3
2
1

Stack after Reversing
1
2
3
4
5
```

Time Complexity: O(N^2)

The time complexity of the above approach to reverse a stack is O(N * N) (where 'N' is the number of elements in the Stack) because we are first removing each element from the Stack and for each element to insert at the bottom we are again removing all the elements.

Space Complexity: O(1)

The space complexity for the above approach to reverse a stack is O(1) because we are not using any auxiliary space.

Python program to reverse a string using stack

```
class  Streverse:
    def __init__(self):
        self.items=list()
        self.size=-1

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

    def  pop(self):
        if  self.isEmpty():
            print("Stack is empty")
        else:
            return self.items.pop()
            self.size-=1

    def  push(self,item):
        self.items.append(item)
        self.size+=1

    def reverse(self,string):
        n=len(string)

        for i in range(0,n):
            S.push(string[i])

        string=""
```

```
    for i in range(0,n):
        string+=S.pop()
    return string

S=Streverse()
strname=input("Enter a string to be reversed:")
strrev=S.reverse(strname)
print("Reversed string is " +strrev)
```

OUTPUT

```
Enter a string to be reversed: PYTHON PROGRAM
Reversed string is MARGORP NOHTYP
```

PROGRAM-10

Implement python code to evaluate a Expression tree.

```
class node:
    def __init__(self, value):
        self.left = None
        self.data = value
        self.right = None

def evaluateExpTree(root):
    if root is None:
        return 0

    if root.left is None and root.right is None:
        return int(root.data)

    left_sum = evaluateExpTree(root.left)
     right_sum = evaluateExpTree(root.right)

    if root.data == '+':
        return left_sum + right_sum

    elif root.data == '-':
        return left_sum - right_sum

    elif root.data == '*':
        return left_sum * right_sum

    else:
        return left_sum / right_sum

if __name__=='__main__':
```

```
root = node('+')
root.left = node('*')
root.left.left = node('5')
root.left.right = node('4')
root.right = node('-')
root.right.left = node('100')
root.right.right = node('20')
print("The Output of a first tree is")
print (evaluateExpTree(root))

root = None
root = node('+')
root.left = node('*')
root.left.left = node('5')
root.left.right = node('4')
root.right = node('-')
root.right.left = node('100')
root.right.right = node('/')
root.right.right.left = node('20')
root.right.right.right = node('2')
print("The Output of a second tree is")
print (evaluateExpTree(root))

root = node('+')
root.left = node('*')
root.left.left = node('5')
root.left.right = node('-4')
root.right = node('-')
root.right.left = node('100')
root.right.right = node('20')
print("The Output of a third tree is")
print (evaluateExpTree(root))
```

OUTPUT

The Output of a first tree is

100

The Output of a second tree is

110.0

The Output of a third tree is

60