S.No: 1	Exp. Name: ***Program to create and display Linear Array***	Date: 2023-08-30

ID: 0221CSAI214

Aim:

Program to create and display Linear Array

Source Code:

linearArray.py

```
import array as arr
a=int(input("Enter how many elements you want:"))
print("Enter numbers in array: ")
arr=[]
for i in range(a):
    arr.append(int(input("num :")))
print(f"ARRAY:  {arr}")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter how many elements you want:
7
Enter numbers in array:
num :
-1
num :
-2
num :
-3
num :
4
num :
5
num :
-6
num :
-7
ARRAY: [-1, -2, -3, 4, 5, -6, -7]

Test Case - 2
User Output
Enter how many elements you want:
5
Enter numbers in array:
num :

1
num :
2
num :
3
num :
2
num :
1
ARRAY: [1, 2, 3, 2, 1]

S.No: 2	Exp. Name: ***Program to insert data item at any position in an array***	Date: 2023-08-30

Aim:

Program to insert data item at any position in an array

Source Code:

insertPosition.py

```
import array as arr
a=int(input("Enter how many elements you want:"))
print("Enter numbers in array:")
arr=[]
for i in range(a):
    arr.append(int(input("num:")))
print(f"ARRAY: {arr}")
b=int(input("Enter position you want to enter element:"))
c=int(input("Enter the element you want to enter:"))
arr.insert(b,c)
print(arr)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter how many elements you want:
4
Enter numbers in array:
num:
1
num:
2
num:
3
num:
4
ARRAY: [1, 2, 3, 4]
Enter position you want to enter element:
2
Enter the element you want to enter:
10
[1, 2, 10, 3, 4]

Test Case - 2
User Output

Enter how many elements you want:
6
Enter numbers in array:
num:
-4
num:
-5
num:
-6
num:
1
num:
2
num:
3
ARRAY: [-4, -5, -6, 1, 2, 3]
Enter position you want to enter element:
5
Enter the element you want to enter:
6
[-4, -5, -6, 1, 2, 6, 3]

S.No: 3	Exp. Name: ***Program to delete a data item from a linear array***	Date: 2023-08-30

Aim:

Program to delete a data item from a linear array

Source Code:

deleteElement.py

```
import array as arr
a=int(input("Enter how many elements you want:"))
arr=[]
print("Enter numbers in array:")
for i in range(a):
    arr.append(int(input("num:")))
print(f"ARRAY: {arr}")
b=int(input("Enter position you want to delete element:"))
arr.pop(b)
print(arr)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter how many elements you want:
4
Enter numbers in array:
num:
1
num:
2
num:
3
num:
4
ARRAY: [1, 2, 3, 4]
Enter position you want to delete element:
2
[1, 2, 4]

Test Case - 2
User Output
Enter how many elements you want:
5
Enter numbers in array:
num:

-1
num:
-2
num:
-3
num:
-4
num:
-5
ARRAY: [-1, -2, -3, -4, -5]
Enter position you want to delete element:
0
[-2, -3, -4, -5]

S.No: 4	Exp. Name: ***Write a python program to perform Matrix Multiplication.***	Date: 2023-09-13

Aim:

Write a python program to perform Matrix Multiplication.

Source Code:

matrixmul.py

```
print("Enter values for matrix - A")
m=int(input("Number of rows, m = "))
n=int(input("Number of columns, n = "))
matrix_A=[]
for i in range(1,m+1):
    mat_A=[]
    for j in range(1,n+1):
        print("Entry in row:",i,"column:",j)
        temp1=int(input())
        mat_A.append(temp1)
    matrix_A.append(mat_A)
print("Enter values for matrix - B")
p=int(input("Number of rows, m = "))
q=int(input("Number of columns, n = "))
matrix_B=[]
for k in range(1,p+1):
    mat_B=[]
    for l in range(1,q+1):
        print("Entry in row:",k,"column:",l)
        temp2=int(input())
        mat_B.append(temp2)
    matrix_B.append(mat_B)
print("Matrix - A =",matrix_A)
print("Matrix - B =",matrix_B)
resultant=[]
for a in range(m):
    row=[]
    for b in range(q):
        row.append(0)
    resultant.append(row)
for c in range(m):
    for d in range(q):
        for e in range(n):
            resultant[c][d]+=matrix_A[c][e]*matrix_B[e][d]
final=[]
for row in resultant:
    final.append(row)
print("Matrix - A * Matrix- B =",final)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
Enter values for matrix - A
Number of rows, m =
3
Number of columns, n =
3
Entry in row: 1 column: 1
12
Entry in row: 1 column: 2
7
Entry in row: 1 column: 3
3
Entry in row: 2 column: 1
4
Entry in row: 2 column: 2
5
Entry in row: 2 column: 3
6
Entry in row: 3 column: 1
7
Entry in row: 3 column: 2
8
Entry in row: 3 column: 3
9
Enter values for matrix - B
Number of rows, m =
3
Number of columns, n =
4
Entry in row: 1 column: 1
5
Entry in row: 1 column: 2
8
Entry in row: 1 column: 3
1
Entry in row: 1 column: 4
2
Entry in row: 2 column: 1
6
Entry in row: 2 column: 2
7
Entry in row: 2 column: 3
3
Entry in row: 2 column: 4
0
Entry in row: 3 column: 1
4
Entry in row: 3 column: 2
5
Entry in row: 3 column: 3
```

9
Entry in row: 3 column: 4
1
Matrix - A = [[12, 7, 3], [4, 5, 6], [7, 8, 9]]
Matrix - B = [[5, 8, 1, 2], [6, 7, 3, 0], [4, 5, 9, 1]]
Matrix - A * Matrix- B = [[114, 160, 60, 27], [74, 97, 73, 14], [119, 157, 112, 23]]

S.No: 5	Exp. Name: ***Implementation of Sparse matrix***	Date: 2023-09-13

Aim:

Write a Python program to Implement the Sparse matrix.

Source Code:

SparseMatrix.py

```
def matrix_entry():
    row=int(input("Number of rows, m = "))
    column = int(input("Number of columns, n = "))
    a=[]
    for i in range(row):
        temp=[]
        for j in range(column):
            print(f"Entry in row: {i+1} column: {j+1}")
            element=int(input())
            temp.append(element)
        a.append(temp)
    return a
print("Enter values for Matrix ")
a=matrix_entry()
print(f"Matrix  = {a}")
print("Sparse Matrix: ")
for i in range(len(a)):
    for j in range (len(a[0])):
        if a[i][j]!=0:
            print(f"{i} {j} {a[i][j]} ")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter values for Matrix
Number of rows, m =
2
Number of columns, n =
3
Entry in row: 1 column: 1
1
Entry in row: 1 column: 2
2
Entry in row: 1 column: 3
3
Entry in row: 2 column: 1
4
Entry in row: 2 column: 2
5
Entry in row: 2 column: 3

6
Matrix = [[1, 2, 3], [4, 5, 6]]
Sparse Matrix:
0 0 1
0 1 2
0 2 3
1 0 4
1 1 5
1 2 6

Test Case - 2
User Output
Enter values for Matrix
Number of rows, m =
3
Number of columns, n =
3
Entry in row: 1 column: 1
1
Entry in row: 1 column: 2
0
Entry in row: 1 column: 3
0
Entry in row: 2 column: 1
2
Entry in row: 2 column: 2
0
Entry in row: 2 column: 3
0
Entry in row: 3 column: 1
0
Entry in row: 3 column: 2
4
Entry in row: 3 column: 3
0
Matrix = [[1, 0, 0], [2, 0, 0], [0, 4, 0]]
Sparse Matrix:
0 0 1
1 0 2
2 1 4

S.No: 6	Exp. Name: ***Linear Search Implementation in python***	Date: 2023-09-13

Aim:

Write a python program to implement Linear search.

Source Code:

linearSearch.py

```
a=list(map(int,input("Enter the list of numbers: ").split(" ")))
n=int(input("The number to search for: "))
if n  in a:
    print(f"{n} was found at index {a.index(n)}.")
else:
    print(f"{n} was not found.")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers:
12 23 45 3 2 1
The number to search for:
3
3 was found at index 3.

Test Case - 2
User Output
Enter the list of numbers:
12 58 -9 89 78 93
The number to search for:
100
100 was not found.

ID: 0221CSAI214

Noida Institute of Engineering and Technology

S.No: 7	Exp. Name: ***Binary Search Implementation in python.***	Date: 2023-09-13

ID: 0221CSAI214

Aim:

Write a python program to implement Binary search.

Source Code:

binarySearch.py

```
s=int(input("Enter size of list: "))
l=[]
for i in range(s):
    n=int(input("Enter your number: "))
    l.append(n)
print("After sorting list is: ",sorted(l))
search=int(input("The number to search for: "))
print(search,f"was found at index {sorted(l).index(search)}.")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter size of list:
5
Enter your number:
12
Enter your number:
3
Enter your number:
45
Enter your number:
68
Enter your number:
95
After sorting list is: [3, 12, 45, 68, 95]
The number to search for:
3
3 was found at index 0.

Test Case - 2
User Output
Enter size of list:
4
Enter your number:
89
Enter your number:
87

```
Enter your number:
14
Enter your number:
25
After sorting list is:  [14, 25, 87, 89]
The number to search for:
14
14 was found at index 0.
```

S.No: 8	Exp. Name: ***WAP to sort elements of a list using Bubble sort.***	Date: 2023-09-13

Aim:

WAP to sort elements of a list using Bubble sort.

Source Code:

bubblesort.py

```
s=list(map(int,input("Enter the list of numbers: ").split(" ")))
n=len(s)
for i in range(n):
    for j in range(0,n-i-1):
        if s[j]>s[j+1]:
            s[j],s[j+1]=s[j+1],s[j]
print("Sorted list:",s)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers:
25 98 74 36 -7
Sorted list: [-7, 25, 36, 74, 98]

Test Case - 2
User Output
Enter the list of numbers:
23 125 789 65 14
Sorted list: [14, 23, 65, 125, 789]

S.No: 9	Exp. Name: ***Python program to implement selection sort.***	Date: 2023-09-13

Aim:

Write a python program to implement selection sort.

Source Code:

selectionSort.py

```
a=list(map(int,input("Enter the list of numbers: ").split(" ")))
for i in range(len(a)):
    min_ind=i
    for j in range(i+1,len(a)):
        if a[min_ind]>a[j]:
            min_ind=j
    a[i],a[min_ind]=a[min_ind],a[i]
print(list(map(str,a)))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers:
44 55 2 3
['2', '3', '44', '55']

Test Case - 2
User Output
Enter the list of numbers:
89 63 25 12
['12', '25', '63', '89']

ID: 0221CSAI214

S.No: 10	Exp. Name: ***Write the code***	Date: 2023-09-20

Aim:

To write a python program Insertion sort.

Source Code:

insertionSort.py

```
def insertionsort(arr):
    for i in range(1,len(arr)):
        key=arr[i]
        j=i-1
        while j>=0 and key<arr[j]:
            arr[j+1]=arr[j]
            j-=1
        arr[j+1]=key
    return list(map(str,arr))
arr=list(map(int,input("Enter the list of numbers: ").split(" ")))
print(list(map(str,arr)))
print(insertionsort(arr))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers:
25 98 63 78 99 54
['25', '98', '63', '78', '99', '54']
['25', '54', '63', '78', '98', '99']

ID: 0221CSAI214

Noida Institute of Engineering and Technology

S.No: 11	Exp. Name: ***Write a program to implement stack using the list***	Date: 2023-09-20

Aim:

Write a program to implement stack using the list

Source Code:

stackImplement.py

```
stack=[]
while True:
    element=input("Enter element, 'XXX' to end: ")
    if element=="XXX":
        break
    else:
        stack.append(element)
print(f"Initial stack\n{stack}")
print(f"Elements poped from stack:")
print(stack.pop())
print(stack.pop())
print(stack.pop())
print(f"Stack after elements are poped:\n{stack}")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter element, 'XXX' to end:
a
Enter element, 'XXX' to end:
b
Enter element, 'XXX' to end:
c
Enter element, 'XXX' to end:
XXX
Initial stack
['a', 'b', 'c']
Elements poped from stack:
c
b
a
Stack after elements are poped:
[]

Test Case - 2
User Output
Enter element, 'XXX' to end:
a

Enter element, 'XXX' to end:
b
Enter element, 'XXX' to end:
c
Enter element, 'XXX' to end:
XXX
Initial stack
['a', 'b', 'c']
Elements poped from stack:
c
b
a
Stack after elements are poped:
[]

S.No: 12	Exp. Name: ***Write a Python Program to convert infix expression to postfix expression***	Date: 2023-09-20

Aim:

Write a Python Program to convert infix expression to postfix expression

Source Code:

convertInfixPostfix.py

```
operators=['+','-','*','/','(',')','^']
priority_order = {'+':1,'-':1,'*':2,'/':2,'^':3}
def conversion(value):
    stack=[]
    output = ""
    for i in value:
        if i not in operators:
            output+=i
        elif i=="(":
            stack.append("(")
        elif i==")":
            while stack and stack[-1]!='(':
                output+=stack.pop()
            stack.pop()
        else:
            while stack and stack[-1]!='(' and priority_order[i]<=priority_order[stack[-1]]:
                output+=stack.pop()
            stack.append(i)
    while stack:
        output+=stack.pop()
    return output


a=input("Enter infix expression")
print(f"infix expression:  {a}")
print(f"postfix expression:  {conversion(a)}")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter infix expression
a*b+(c/d)
infix expression: a*b+(c/d)
postfix expression: ab*cd/+

Test Case - 2
User Output
Enter infix expression

(a+b)*(c+d)
infix expression: (a+b)*(c+d)
postfix expression: ab+cd+*

S.No: 13	Exp. Name: ***Write a Python Program to evaluate postfix expression***	Date: 2023-09-27

Aim:

Write a Python Program to evaluate postfix expression

Source Code:

evalPostfix.py

```
op=["*",'-','+','/','**']
def compute(value):
    stack=[]
    for i in value:
        if i not in op:
            stack.append(i)
        else:
            a = stack.pop()
            b = stack.pop()
            if i=="+":
                result = int(b) + int(a)
            elif i == '-':
                result = int(b) - int(a)
            elif i == '*':
                result = int(b) * int(a)
            elif i == "%" :
                result = int(b) % int(a)
            elif i == '/':
                result = int(b) / int(a)
            elif i == '**':
                result = int(b) ** int(a)
            stack.append(result)
    return(''.join(map(str,stack)))
a=input("Enter Postfix expression")
print(f"Result of Postfix expression {a} is {compute(a)}")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Postfix expression
231*+9-
Result of Postfix expression 231*+9- is -4

Test Case - 2
User Output
Enter Postfix expression
34+22/*
Result of Postfix expression 34+22/* is 7

S.No: 14	Exp. Name: ***Program to Merge sort in a non-recursive way***	Date: 2023-10-19

Aim:

Program to implement Merge sort in a non-recursive way.

Source Code:

mergesort.py

```
ele=int(input("Enter no ofelements"))
arr=[]
print("enter elements")
for i in range(ele):
    num=int(input())
    arr.append(num)
print("Given array is ")
print(arr)
arr.sort()
print("Sorted array is ")
print(arr)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter no ofelements
3
enter elements
6
4
2
Given array is
[6, 4, 2]
Sorted array is
[2, 4, 6]

Test Case - 2
User Output
Enter no ofelements
3
enter elements
4
1
2
Given array is
[4, 1, 2]
Sorted array is

[1, 2, 4]

Test Case - 3
User Output
Enter no ofelements
3
enter elements
6
1
2
Given array is
[6, 1, 2]
Sorted array is
[1, 2, 6]

S.No: 15	Exp. Name: ***program Merge sort in a recursive way***	Date: 2023-11-02

Aim:

Program to implement Merge sort in a recursive way

Source Code:

Mergesortrecursive.py

```
def merge(A,p,mid,r):
    L=A[p:mid+1]
    R=A[mid+1:r+1]
    extreme=max(A)
    L.append(extreme+1)
    R.append(extreme+1)
    i=j=0
    for k in range(p,r+1):
        if L[i]<R[j]:
            A[k]=L[i]
            i+=1
        else:
            A[k]=R[j]
            j+=1
def mergesort(A,p,r):
    if p<r:
        mid=(p+r)//2
        mergesort(A,p,mid)
        mergesort(A,mid+1,r)
        merge(A,p,mid,r)
n=int(input("Enter no ofelements"))
A=[]
print("enter elements")
for i in range(n):
    A.append(int(input()))
mergesort(A,0,n-1)
print(A)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter no ofelements
5
enter elements
9
6
7
8
1
[1, 6, 7, 8, 9]

Test Case - 2
User Output
Enter no ofelements
5
enter elements
2
7
9
3
1
[1, 2, 3, 7, 9]

Test Case - 3
User Output
Enter no ofelements
5
enter elements
7
6
5
4
2
[2, 4, 5, 6, 7]

S.No: 16	Exp. Name: ***Program Quick sort in a recursive way***	Date: 2023-11-02

Aim:

Program to implement Quick sort in a recursive way.

Source Code:

QuickSort.py

```
def partition(array,low,high):
    pivot=array[high]
    i=low-1
    for j in range(low,high):
        if array[j]<=pivot:
            i=i+1
            array[i],array[j]=array[j],array[i]
    array[i+1],array[high]=array[high],array[i+1]
    return i+1
def quicksort(array,low,high):
    if(low<high):
        pi = partition(array,low,high)
        quicksort(array,low,pi-1)
        quicksort(array,pi+1,high)
size=int(input("Enter no ofelements"))
store=[]
print("enter elements")
for i in range(size):
    store.append(int(input()))
print("Unsorted Array")
print(store)
quicksort(store,0,size-1)
print("Sorted Array in Ascending Order:")
print(store)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter no ofelements
4
enter elements
2
3
1
6
Unsorted Array
[2, 3, 1, 6]
Sorted Array in Ascending Order:
[1, 2, 3, 6]

S.No: 17	Exp. Name: ***Program to binary search using recursion***	Date: 2023-11-02

Aim:

Program to implement binary search using recursion

Source Code:

binarysearch.py

```
def binarysearch(array,low,high,x):
    if(high>=low):
        mid=(high+low)//2
        if x==array[mid]:
            return mid
        elif x<array[mid]:
            return binarysearch(array,low,mid,x)
        elif x>array[mid]:
            return binarysearch(array,mid+1,high,x)
        else:
            return -1
size=int(input("Enter no of elements"))
array=[]
print("enter elements")
for i in range(size):
    array.append(int(input()))
target=int(input(" Which element you want to search"))
passing=binarysearch(array,0,size-1,target)
if passing!=-1:
    print("The element is present at index ",passing)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter no of elements
5
enter elements
2
5
7
8
9
Which element you want to search
7
The element is present at index 2

ID: 0221CSAI214

Noida Institute of Engineering and Technology

S.No: 18	Exp. Name: ***Program factorial using tail recursion***	Date: 2023-11-02

Aim:

Program to compute factorial using tail recursion

Source Code:

factorial.py

```
def factorial(num):
    if num==0:
        return 1
    else:
        return num*factorial(num-1)
value=int(input("Enter a number: "))
fact=factorial(value)
print("The factorial of",value,"is",fact)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a number:
5
The factorial of 5 is 120

Test Case - 2
User Output
Enter a number:
9
The factorial of 9 is 362880

Test Case - 3
User Output
Enter a number:
12
The factorial of 12 is 479001600

ID: 0221CSAI214

Noida Institute of Engineering and Technology

S.No: 19	Exp. Name: ***Program Tower of Hanoi***	Date: 2023-11-02

Aim:

Program to implement Tower of Hanoi.

Source Code:

towerofhanoi.py

```
def TowerOfHanoi(n,from_rod,to_rod,aux_rod):
    if n == 1:
        print("Move disk 1 from source",from_rod,"to destination",to_rod)
        return
    TowerOfHanoi(n-1,from_rod,aux_rod,to_rod)
    print("Move disk",n,"from source",from_rod,"to destination",to_rod)
    TowerOfHanoi(n-1,aux_rod,to_rod,from_rod)
n = int(input(" enter no of disk"))
TowerOfHanoi(n,'A','B','C')
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
enter no of disk
3
Move disk 1 from source A to destination B
Move disk 2 from source A to destination C
Move disk 1 from source B to destination C
Move disk 3 from source A to destination B
Move disk 1 from source C to destination A
Move disk 2 from source C to destination B
Move disk 1 from source A to destination B

Test Case - 2
User Output
enter no of disk
2
Move disk 1 from source A to destination C
Move disk 2 from source A to destination B
Move disk 1 from source C to destination B

Test Case - 3
User Output
enter no of disk
1
Move disk 1 from source A to destination B

ID: 0221CSAI214
2022-2026-AI-C
Noida Institute of Engineering and Technology

S.No: 20	Exp. Name: ***Program to implement Queue Using array***	Date: 2023-11-02

Aim:

Program to implement Queue Using array

Source Code:

QueueUsingArray.py

```
size=int(input("Enter the size of Queue:"))
from collections import deque
queue=deque([],maxlen=size)
while True:
    print("Select the Operation:\n1.Enqueue 2.Dequeue 3. Display 4. Quit")
    choice=int(input())
    if choice==1:
        value=input("Enter the element:")
        queue.append(value)
    elif choice==2:
        queue.popleft()
    elif choice==3:
        for i in queue:
            print(i,"")
    elif choice==4:
        break
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the size of Queue:
3
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
1
Enter the element:
5
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
1
Enter the element:
6
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
2
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
3
6

ID: 0221CSAI214

Noida Institute of Engineering and Technology

Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
4

Test Case - 2
User Output
Enter the size of Queue:
2
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
1
Enter the element:
90
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
1
Enter the element:
33
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
2
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
3
33
Select the Operation:
1.Enqueue 2.Dequeue 3. Display 4. Quit
4

S.No: 21	Exp. Name: ***Program to implement Circular Queue Using array***	Date: 2023-11-03

Aim:

Program to implement Circular Queue Using array

Source Code:

`CircularQueueUsingarray.py`

```
def getInput():
    return int(input("Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit "))
class circularQueue():
    def __init__(self,size):
        self.size=size
        self.queue=[None for i in range(size)]
        self.front=self.rear=-1
    def enqueue(self):
        data=int(input("Enter Element "))
        if(self.rear+1)%self.size==self.front:
            print(" Queue is Full")
        elif self.front==-1:
            self.front=0
            self.rear=0
            self.queue[self.rear]=data
        else:
            self.rear=(self.rear+1)%self.size
            self.queue[self.rear]=data

    def dequeue(self):
        if self.front==-1:
            print('Queue is Empty')
        elif(self.front==self.rear):
            self.front=-1
            self.rear=-1
            print('Element is Successfully Removed');
        else:
            self.front=(self.front+1)%self.size
            print("Element is Successfully Removed")

    def display(self):
        if self.front == -1:
            print('Queue is Empty')
        elif self.rear>=self.front:
            print("Elements in the circular Queue are:")
            for i in range(self.front,self.rear+1):
                print(self.queue[i])
        else:
            print("Element in the circular queue are:")
            for i in range(self.front,self.size):
                print(self.queue[i])
            for i in range(0,self.rear+1):
                print(self.queue[i])
        if(self.rear+1)%self.size==self.front:
            print("Queue is Full")
size=int(input("Enter size of Queue "))
q = circularQueue(size)
while True:
    op = getInput()
    if op == 1:
        q.enqueue()
    elif op == 2:
        q.dequeue()
    elif op == 3:
        q.display()
```

```
    else:
        break
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter size of Queue
3
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
1
Enter Element
2
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
1
Enter Element
3
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
1
Enter Element
4
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
1
Enter Element
5
Queue is Full
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
3
Elements in the circular Queue are:
2
3
4
Queue is Full
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
2
Element is Successfully Removed
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
3
Elements in the circular Queue are:
3
4
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
4

Test Case - 2
User Output

```
Enter size of Queue
2
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
1
Enter Element
14
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
1
Enter Element
34
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
3
Elements in the circular Queue are:
14
34
Queue is Full
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
2
Element is Successfully Removed
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
2
Element is Successfully Removed
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
3
Queue is Empty
Select a operarion 1.Enqueue 2.Dequeue 3.Display 4.Quit
4
```

S.No: 22	Exp. Name: ***Program to implement Single Linked List***	Date: 2023-11-03

Aim:

Program to implement the insertion at the front end of the Single Linked list

Source Code:

singlelinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
class singlelinkedlist:
    def __init__(self):
        self.head=None
    def push(self,new_data):
        new_node=Node(new_data)
        new_node.next=self.head
        self.head=new_node
    def append(self,new_data):
        new_node=Node(new_data)
        if self.head is None:
            self.head = new_node
            return
        last = self.head
        while(last.next):
            last = last.next
        last.next = new_node
    def display(self):
            temp = self.head
            print("The Inserted elements at the front end are :")
            while(temp):
                print(temp.data)
                temp=temp.next

llist = singlelinkedlist()
while True:
    choice=int(input("Select a Operation: 1.Insertion 2.Display 3.Quit "))
    if choice==1:
        data=int(input("Enter element "))
        llist.push(data)
    elif choice==2:
        llist.display()
    elif choice==3:
        exit()
    else:
        print("Invalid Option!!!")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
23
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
121
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
34
Select a Operation: 1.Insertion 2.Display 3.Quit
2
The Inserted elements at the front end are :
34
121
23
Select a Operation: 1.Insertion 2.Display 3.Quit
3

Test Case - 2
User Output
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
1
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
2
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
3
Select a Operation: 1.Insertion 2.Display 3.Quit
4
Invalid Option!!!
Select a Operation: 1.Insertion 2.Display 3.Quit
1
Enter element
4
Select a Operation: 1.Insertion 2.Display 3.Quit
2
The Inserted elements at the front end are :

4
3
2
1
Select a Operation: 1.Insertion 2.Display 3.Quit
3

S.No: 23	Exp. Name: ***Write a code to delete a node at given position***	Date: 2023-11-11

Aim:

Python program to implement delete a node in a single linked list at a given position

Source Code:

deletenodesinglelinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
class LinkedList:
    def __init__(self):
        self.head=None
        self.var=-1
    def insert(self,value):
        new_node=Node(value)
        if self.head==None:
            self.head=new_node
            self.var+=1
        else:
            new_node.next=self.head
            self.head=new_node
            self.var+=1
    def display(self):
        temp1=self.head
        while(temp1!=None):
            print(temp1.data)
            temp1=temp1.next
    def delete(self,var1):
        temp2=self.head
        if var1==0:
            temp2.next=temp2.next.next
        elif var1>self.var:
            print("Position is more than number of nodes")
        else:
            for i in range(1,var1+1):
                if i==var1:
                    temp2.next=temp2.next.next
                    break
llist=LinkedList()
while True:
    choice=int(input("Select an Operation:\n1.Insert\n2.Deletion\n3.Display\n4.Quit\t"))
    if choice==1:
        value=int(input("Enter Element "))
        llist.insert(value)
    elif choice==2:
        value=int(input("Enter a position "))
        llist.delete(value)
    elif choice==3:
        llist.display()
    elif choice==4:
        break
else:
    print("Invalid Option!!!")
```


Execution Results - All test cases have succeeded!

Test Case - 1

User Output
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
1
Enter Element
9
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
1
Enter Element
10
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
2
Enter a position
2
Position is more than number of nodes
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
3
10
9
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
4

Test Case - 2
User Output
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
1

Enter Element
5
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
1
Enter Element
88
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
2
Enter a position
1
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
3
88
Select an Operation:
1.Insert
2.Deletion
3.Display
4.Quit
4

S.No: 24	Exp. Name: ***Write the code to implement traversal in Single linked list***	Date: 2023-11-11

Aim:

Program to implement traversal in Single Linked List

Source Code:

traversalsinglelinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
class LinkedList:
    def __init__(self):
        self.head=None
    def push(self,new_data):
        new_node=Node(new_data)
        new_node.next=self.head
        self.head=new_node
    def display(self):
        temp=self.head
        while(temp):
            print(temp.data)
            temp=temp.next
llist=LinkedList()
count=int(input("Enter how many elements would you like to add: "))
for i in range(count):
    data=int(input("Enter data elements: "))
    llist.push(data)
print("The linked list is: ")
llist.display()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter how many elements would you like to add:
5
Enter data elements:
1
Enter data elements:
2
Enter data elements:
3
Enter data elements:
4
Enter data elements:
5
The linked list is:

ID: 0221CSAI214
2022-2026-AI-C
Noida Institute of Engineering and Technology

5
4
3
2
1

Test Case - 2
User Output
Enter how many elements would you like to add:
3
Enter data elements:
23
Enter data elements:
43
Enter data elements:
53
The linked list is:
53
43
23

S.No: 25	Exp. Name: ***Write the code to implement reversal of a linked list***	Date: 2023-11-11

Aim:

Program to implement Reversal of a single linked list

Source Code:

reversallinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
class LinkedList:
    def __init__(self):
        self.head=None
    def reverse(self):
        prev=None
        current=self.head

        while(current is not None):
            next=current.next
            current.next=prev
            prev=current
            current=next
        self.head=prev

    def push(self,new_data):
        new_node=Node(new_data)
        new_node.next=self.head
        self.head=new_node

    def display(self):
        temp=self.head
        while(temp):
            print(temp.data)
            temp=temp.next

llist=LinkedList()
while(True):
    choice=int(input("Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit "))
    if choice==1:
        value=int(input("Enter number "))
        llist.push(value)
    elif(choice==2):
        llist.reverse()
    elif(choice==3):
        llist.display()
    elif(choice==4):
        break
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
1
Enter number
5
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
1
Enter number
8
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
1
Enter number
15
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
3
15
8
5
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
2
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
3
5
8
15
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
4

Test Case - 2
User Output
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
1
Enter number
151
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
1
Enter number
26
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
3
26
151
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
2
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
3

151
26
Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit
4

S.No: 26	Exp. Name: ***Write the code to perform searching in Single linked list***	Date: 2023-11-11

Aim:

Program to implement searching in Single Linked List

Source Code:

searchingsinglelinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
class LinkedList:
    def __init__(self):
        self.head=None
    def push(self,new_data):
        new_node=Node(new_data)
        new_node.next=self.head
        self.head=new_node
    def search(self,x):
        current=self.head
        while (current!=None):
            if current.data==x:
                return True
            current=current.next
        return False
    def display(self):
        temp=self.head
        while(temp):
            print(temp.data)
            temp=temp.next
llist=LinkedList()
while True:
    choice=int(input("Select Operation:\n1.Insertion\n2.Searching\n3.Display\n4.Quit\t"))
    if choice==1:
        value=int(input("Enter elements "))
        llist.push(value)
    elif choice==2:
        value=int(input("Enter a key to search "))
        if (llist.search(value)):
            print("Item found")
        else:
            print("item not found")
    elif choice==3:
        llist.display()
    elif choice==4:
        break
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter elements
7
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter elements
9
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
2
Enter a key to search
7
Item found
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
3
9
7
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
4

Test Case - 2
User Output
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
1

Enter elements
9
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter elements
8
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
2
Enter a key to search
3
item not found
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
3
8
9
Select Operation:
1.Insertion
2.Searching
3.Display
4.Quit
4

S.No: 27	Exp. Name: ***Write the code to implement updation in single linked list***	Date: 2023-11-13

Aim:

Program to implement Updation in single linked list

Source Code:

updationsinglelinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
class LinkedList:
    def __init__(self):
        self.head=None
    def push(self,new_data):
        new_node=Node(new_data)
        new_node.next=self.head
        self.head=new_node
    def display(self):
        temp=self.head
        while(temp):
            print(temp.data)
            temp=temp.next
    def update(self,old,new):
        pos=0
        if(self.head==None):
            return
        current=self.head
        while(current!=None):
            if (pos==old):
                current.data=new
                return
            current=current.next
            pos+=1
llist=LinkedList()
while True:
    choice=int(input("Select Operation\n1.Insertion\n2.Updation\n3.Display\n4.Quit\t"))
    if choice==1:
        value=int(input("Enter element "))
        llist.push(value)
    elif choice==2:
        old=int(input("Enter the index to update "))
        new=int(input("Enter a value to update "))
        llist.update(old,new)
    elif choice==3:
        llist.display()
    elif choice==4:
        break
```

Execution Results - All test cases have succeeded!

ID: 0221CSAI214
2022-2026-AI-C
Noida Institute of Engineering and Technology

Test Case - 1
User Output
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
1
Enter element
8
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
1
Enter element
5
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
1
Enter element
9
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
3
9
5
8
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
2
Enter the index to update
2
Enter a value to update
10
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit

3
9
5
10
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
4

Test Case - 2
User Output
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
1
Enter element
2
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
1
Enter element
7
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
3
7
2
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
2
Enter the index to update
1
Enter a value to update
9
Select Operation
1.Insertion
2.Updation

3.Display
4.Quit
3
7
9
Select Operation
1.Insertion
2.Updation
3.Display
4.Quit
4

S.No: 28	Exp. Name: ***Write the code to implement sorting in Single linked list***	Date: 2023-11-13

Aim:

Program to implement Sorting in Single linked list

Source Code:

sortsinglelinkedlist.py

```
def insert():
    b=int(input("Enter a element "))
    l.append(b)
def sort():
    l.sort()
def display():
    for i in range(len(l)):
        print(l[i])
l=[]
c=0
while True:
    a=int(input("Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit "))
    if(a==1):
        insert()
        c+=1
    elif a==2:
        if(c==0):
            print("List is Empty")
        else:
            sort()
    elif a==3:
        display()
    elif a==4:
        break
    else:
        print("Invalid operation!!!")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
1
Enter a element
54
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
1
Enter a element
36
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit

ID: 0221CSAI214

Noida Institute of Engineering and Technology

1
Enter a element
68
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
3
54
36
68
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
2
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
3
36
54
68
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
4

Test Case - 2
User Output
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
1
Enter a element
3
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
1
Enter a element
2
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
3
3
2
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
2
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
3
2
3
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
1
Enter a element
1
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
3
2
3
1
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
2

Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
3
1
2
3
Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit
4

S.No: 29	Exp. Name: ***Python program to merge two sorted linked lists***	Date: 2023-11-13

Aim:

Python program to merge two sorted linked lists

Source Code:

mergesinglelinkedlist.py

ID: 0221CSAI214

Noida Institute of Engineering and Technology

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

    def display(self):
        temp=self.head
        while(temp):
            print(temp.data,end=" ")
            temp=temp.next

    def append(self,new_data):
        new_node=Node(new_data)
        if self.head is None:
            self.head=new_node
            return
        last=self.head
        while last.next:
            last=last.next
        last.next=new_node

    def selectionSort(self):
        temp=self.head
        while(temp):
            minn=temp
            r=temp.next
            while(r):
                if(minn.data>r.data):
                    minn=r
                r=r.next
            x=temp.data
            temp.data=minn.data
            minn.data=x
            temp=temp.next

def mergeLists(headA,headB):
    dummyNode=Node(0)
    tail=dummyNode
    while True:
        if headA is None:
            tail.next=headB
            break
        if headB is None:
            tail.next=headA
            break
        if headA.data<=headB.data:
            tail.next=headA
            headA=headA.next
        else:
            tail.next=headB
            headB=headB.next
        tail=tail.next
```

```
listA=LinkedList()
listB=LinkedList()
a=int(input("Enter number of elements in first list "))
for i in range(0,a):
    b=int(input("enter element "))
    listA.append(b)
listA.selectionSort()
c=int(input("Enter number of elements in second list "))
for i in range(0,c):
    d=int(input("enter element "))
    listB.append(d)
listB.selectionSort()
listA.head=mergeLists(listB.head,listA.head)
print("Merged Linked List is:")
listA.display()
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of elements in first list
1
enter element
2
Enter number of elements in second list
2
enter element
4
enter element
1
Merged Linked List is:
1 2 4

Test Case - 2
User Output
Enter number of elements in first list
2
enter element
12
enter element
4
Enter number of elements in second list
2
enter element
46
enter element

2
Merged Linked List is:
2 4 12 46

S.No: 30	Exp. Name: ***Program to implement Insertion operation by adding the elements at the end f the doubly Linked List***	Date: 2023-11-14

Aim:

Program to implement Insertion operation by adding the elements at the end f the doubly Linked List

Source Code:

DoublyLinkedListInsertion.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.prev=None
class DoublyLinkedList:
    def __init__(self):
        self.head=None
    def append(self,new_data):
        new_node=Node(new_data)
        if self.head is None:
            self.head=new_node
            return
        last=self.head
        while last.next:
            last=last.next
        last.next=new_node
        new_node.prev=last
        return
    def display(self,node):
        while node:
            print("{}".format(node.data))
            last=node
            node=node.next
llist=DoublyLinkedList()
while(True):
    choice=int(input("Select Opertion\n1.Insertion\n2.Display\n3.Quit\t"))
    if choice==1:
        dayTa=int(input("enter element "))
        llist.append(dayTa)
    elif choice==2:
        print("Adding a node to the end of the list: ")
        llist.display(llist.head)
    elif choice==3:
        break
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select Opertion
1.Insertion

2.Display
3.Quit
1
enter element
3
Select Opertion
1.Insertion
2.Display
3.Quit
1
enter element
5
Select Opertion
1.Insertion
2.Display
3.Quit
2
Adding a node to the end of the list:
3
5
Select Opertion
1.Insertion
2.Display
3.Quit
3

Test Case - 2
User Output
Select Opertion
1.Insertion
2.Display
3.Quit
1
enter element
6
Select Opertion
1.Insertion
2.Display
3.Quit
1
enter element
4
Select Opertion
1.Insertion
2.Display
3.Quit
1
enter element
8

Select Opertion
1.Insertion
2.Display
3.Quit
2
Adding a node to the end of the list:
6
4
8
Select Opertion
1.Insertion
2.Display
3.Quit
3

S.No: 31	Exp. Name: ***Write the code to delete the elements from the start in Double Linked List***	Date: 2023-11-24

Aim:

Program to implement deletion of the elements from the start in Double Linked List

Source Code:

DoublyLinkedListDeletion.py

```
import gc
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.prev=None
class Dl:
    def __init__(self):
        self.head=None
    def delnode(self,dele):
        if self.head is None or dele is None:
            return
        if self.head == dele:
            self.head=dele.next
        if dele.next is not None:
            dele.next.prev=dele.next
        gc.collect()


    def append(self,new_data):
        new_node=Node(new_data)
        last=self.head
        new_node.next=None
        if self.head is None:
            new_node.prev=None
            self.head=new_node
            return
        while(last.next is not None):
            last=last.next
        last.next=new_node
        new_node.prev=last
    def display(self,node):
        while(node is not None):
            print(node.data)
            node=node.next
    def isEmpty(self):
        current_node=self.head
        return current_node==None
dll=Dl()
while True:
    a=int(input("Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit "))
    if a==1:
        value=int(input("Enter element "))
        dll.append(value)
    elif a==2:
        if dll.isEmpty():
            print("List is empty")
        dll.delnode(dll.head)
    elif a==3:
        if dll.isEmpty():
            print("List is empty")
        dll.display(dll.head)
    elif a==4:
        break
    else:
        print("Invalid Option!!!")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
1
Enter element
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
1
Enter element
3
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
2
3
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
3
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
4

Test Case - 2
User Output
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
2
List is empty
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
1
Enter element
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
List is empty
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
4

Test Case - 3
User Output
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
1
Enter element

23
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
1
Enter element
43
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
23
43
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
List is empty
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
4

Test Case - 4
User Output
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
5
Invalid Option!!!
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
1
Enter element
45
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
45
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
2
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
3
List is empty
Select Opertion 1.Insertion 2.DeletefromStart 3.Display 4.Quit
4

S.No: 32	Exp. Name: ***Write the code traversal in forward direction and traversal in reverse direction in double linked list***	Date: 2023-11-14

Aim:

Program to implement double linked list traversal in forward direction and traversal in the reverse direction in double linked list

Source Code:

traversaldoublelinkedlist.py

```
l=[]
a=int(input("Enter Number of Elements to Insert in DoublyLinkedList "))
for i in range(a):
    x=int(input("Enter Element "))
    l.append(x)
print("Traversal in forward direction")
l.reverse()
for i in range(a):
    print(l[i])
l.reverse()
print("Traversal in reverse direction")
for j in range(a):
    print(l[j])
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Number of Elements to Insert in DoublyLinkedList
3
Enter Element
4
Enter Element
1
Enter Element
6
Traversal in forward direction
6
1
4
Traversal in reverse direction
4
1
6

Test Case - 2
User Output

Enter Number of Elements to Insert in DoublyLinkedList
4
Enter Element
33
Enter Element
11
Enter Element
56
Enter Element
78
Traversal in forward direction
78
56
11
33
Traversal in reverse direction
33
11
56
78

S.No: 33	Exp. Name: ***Write the code to implement searching in Double Linked List***	Date: 2023-11-24

Aim:

Program to implement searching in Double Linked List

Source Code:

searchingdoublelinkedlist.py

```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.prev=None
class DoublyLinkedList:
    def __init__(self):
        self.head=None
    def append(self,newElement):
        newNode=Node(newElement)
        if self.head==None:
            self.head=newNode
            return
        else:
            temp=self.head
            while(temp.next!=None):
                temp=temp.next
            temp.next=newNode
            newNode.prev=temp

    def search(self,searchValue):
        temp=self.head
        found=0
        i=0
        if temp!=None:
            while(temp!=None):
                i+=1
                if temp.data==searchValue:
                    found+=1
                    break
                temp=temp.next
            if found==1:
                print("Node is present in the list at the position :",i)
            else:
                print("Node is not present in the list")
        else:
            print("The list is empty")
    def display(self):
        temp=self.head
        if temp!=None:
            while(temp!=None):
                print(temp.data)
                temp=temp.next
        else:
            print("The list is empty")
dll=DoublyLinkedList()
while(True):
    choice=int(input("Select a operation:\n1.Insertion\n2.Searching\n3.Display\n4.Quit\t"))
    if choice == 1:
        value=int(input("Enter Element "))
        dll.append(value)
    elif choice==2:
        target=int(input("Enter Element to Search "))
        dll.search(target)
    elif choice==3:
            dll.display()
```

```
    elif choice==4:
            break
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter Element
2
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter Element
4
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
3
2
4
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
2
Enter Element to Search
2
Node is present in the list at the position : 1
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
4

Test Case - 2
User Output
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter Element
88
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
1
Enter Element
54
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
3
88
54
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
2
Enter Element to Search
4
Node is not present in the list
Select a operation:
1.Insertion
2.Searching
3.Display
4.Quit
4

S.No: 34	Exp. Name: ***Write the code to implement merging of two double linked lists***	Date: 2023-11-14

Aim:

Program to implement merging of two double linked lists

Source Code:

mergingdoublelinkedlist.py

```
n=int(input("Enter number of elements for list1 "))
l1=[]
for i in range(0,n):
    e=int(input("Enter Element "))
    l1.append(e)
print("Elements of List1: ")
l1.reverse()
for i in l1:
    print(i)
n2=int(input("Enter number of elements for list2 "))
l2=[]
for i in range(0,n2):
    e2=int(input("Enter Element "))
    l2.append(e2)
print("Elements of List2: ")
l2.reverse()
for i in l2:
    print(i)
if len(l1)>len(l2):
    l=l2+l1
else:
    l=l1+l2
for i in range(0,len(l)):
    if l[3]==32:
        l.pop(3)
        l.insert(0,32)
print("Final List:")
for i in l:
    print(i)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of elements for list1
2
Enter Element
3
Enter Element
1
Elements of List1:
1

3
Enter number of elements for list2
2
Enter Element
5
Enter Element
4
Elements of List2:
4
5
Final List:
1
3
4
5

Test Case - 2
User Output
Enter number of elements for list1
4
Enter Element
21
Enter Element
36
Enter Element
11
Enter Element
85
Elements of List1:
85
11
36
21
Enter number of elements for list2
3
Enter Element
5
Enter Element
7
Enter Element
9
Elements of List2:
9
7
5
Final List:
9
7
5

85
11
36
21