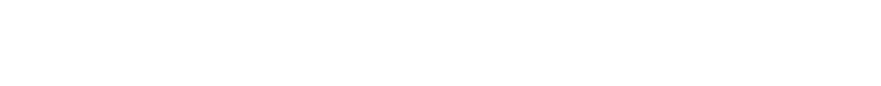
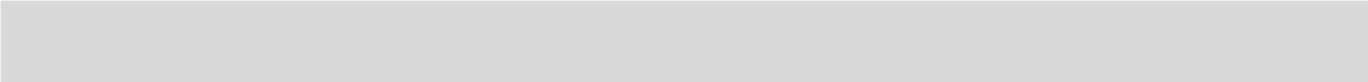
NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY GREATER NOIDA-201306

(An Autonomous Institute)



# DATA STRUCTURES LAB

**Session (2021 – 2022)**

**LAB FILE**

**ON**

**DATA STRUCTURES Using Python Lab**

**(ACSE-0351)**

**(3rd Semester)**

**Submitted To: Submitted By:**

**ALISHA SIKRI Name: SACHIN KUMAR**

**Assistant Professor**

**(AIML) Roll No:2001331530147**

**School of Computer Sciences**

**& Engineering in Emerging**

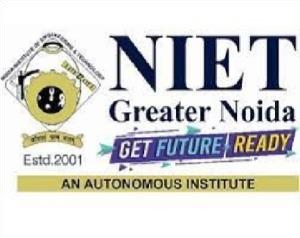
**Technologies**



Affiliated to Dr. A.P.J Abdul Kalam Technical University, Uttar Pradesh, Lucknow.

## NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY GREATER NOIDA-201306

(An Autonomous Institute)

School of Computer Sciences & Engineering in Emerging

Technologies

DATA STRUCTURES Lab (ACSE-0351)

# INDEX

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S.NO | PRACTICAL CONDUCTED | | |  | DATE | PAGE NO | SIGNATURE |
| 1 | Program to create and display Linear Array | | |  |  |  |  |
| 2 | Program to insert a data item at any location in a linear Array | | |  |  |  |  |
| 3 | Program to delete a data item from a Linear Array | | |  |  |  |  |
| 4 | Program to implement multiplication of two matrices | | |  |  |  |  |
| 5 | Program to create sparse matrix. | | |  |  |  |  |
| 6 | Program to implement linear search in an Array. | | |  |  |  |  |
| 7 | Program to implement binary search in an Array. | | |  |  |  |  |
| 8 | Program to implement bubble sort in a non-recursive way. | | |  |  |  |  |
| 9 | Program to implement selection sort in a non-recursive way. | | |  |  |  |  |
| 10 | Program to implement insertion sort in a non-recursive way. | | |  |  |  |  |
| 11 | Write a program to implement stack using the list | | |  |  |  |  |
| 12 |  | Write a Python Program to convert infix expression to | |  |  |  |  |
| postfix expression |  |
| 13 | Write a Python Program to evaluate postfix expressio | | | n |  |  |  |
| 14 | Program to implement Merge sort in a non-recursive way. | | |  |  |  |  |
| 15 | Program to implement Merge sort in a recursive way. | | |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 16 | Program to implement Quick sort in a recursive way. |  |  |  |
| 17 | Program to implement binary search using recursion. |  |  |  |
| 18 | Program to compute factorial using tail recursion. |  |  |  |
| 19 | Program to implement Tower of Hanoi. |  |  |  |
| 20 | Program to implement Queue Using array |  |  |  |
| 21 | Program to implement Circular Queue Using array |  |  |  |
| 22 | Program to implement the Single Linked List  a. Insertion b. Deletion c. Traversal d. Reversal  e. Searching f. Updation g. Sorting h. Merging |  |  |  |
| 23 | Program to implement the doubly Linked List a. Insertion b. Deletion c. Traversal  d. Searching e.Merging |  |  |  |

## Program No: 1 Aim- Program to create and display Linear Array

import array as arr

s=arr.array('i',[])

num1=int(input("Enter how many elements you want:"))

print("Enter numbers in array: ") for i in range(num1):

c=int(input("num :")) s.append(c)

d=list(s)

print("ARRAY: ",d)

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] | | | | | |

Program No: 2 Program to insert data item at any position in an array

import array as arr

s=arr.array('i',[])

num1=int(input("Enter how many elements you want:"))

print("Enter numbers in array:") for i in range(num1):

c=int(input("num:")) s.append(c)

d=list(s) print("ARRAY:",d)

d1=int(input("Enter position you want to enter element:")) d2=int(input("Enter the element you want to enter:")) s.insert(d1,d2) d3=list(s)

print(d3)

Output:

Enter·how·many·elements·you·want:4Enter·numbers·in·array:

num:1 num:2 num:3 num:4

ARRAY:·[1,·2,·3,·4]

Enter·position·you·want·to·enter·element:2

|  |
| --- |
| 10 |

Enter·the·element·you·want·to·enter:

[1,·2,·10,·3,·4]

## Program No: 3 Aim- Program to delete a data item from a linear array

import array as arr

s=arr.array('i',[])

num1=int(input("Enter how many elements you want:"))

print("Enter numbers in array:") for i in range(num1):

c1=int(input("num:")) s.append(c1)

d=list(s) print("ARRAY:",d)

d1=int(input("Enter position you want to delete element:")) s.pop(d1) d3=list(s)

print(d3)

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] | | | | | | |

Program No: 4 Write a python program to perform Matrix Multiplication.

import array as arr

s=arr.array('i',[])

num1=int(input("Enter how many elements you want:"))

print("Enter numbers in array:") for i in range(num1):

c1=int(input("num:")) s.append(c1)

d=list(s) print("ARRAY:",d)

d1=int(input("Enter position you want to delete element:")) s.pop(d1) d3=list(s)

print(d3)

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]] | | | | | | |

Program No: 5 Write a Python program to Implement the Sparse matrix.

print("Enter values for Matrix ") n1=int(input("Number of rows, m = "))

n2=int(input("Number of columns, n = "))

mA=[[0 for i in range(n2)]for j in range(n1)] for i in range(n1): for j in range(n2): print("Entry in row:",i+1,"column:",j+1) c1=int(input())

mA[i][j]=c1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 | | | | | |

print("Matrix =",mA) spare=[] for i in range(len(mA)): for j in range(len(mA[0])): if mA[i][j] !=0: temp=[]

temp.append(i) temp.append(j) temp.append(mA[i][j]) spare.append(temp)

print("Sparse Matrix: ") for i in spare: for j in i:

print(j,end=" ") print()

Output:

|  |  |
| --- | --- |
| 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· | |

Program No: 6 To write a python program linear search.

def search(l,c): for i in range(len(l)): if l[i]==c:

return i

return -1

l=[int(x) for x in input("Enter the list of numbers: ").split()]

c=int(input("The number to search for: ")) result=search(l,c) if result==-1:

print("{} was not found.".format(c)) else:

print("{} was found at index {}.".format(c,result))

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| Enter·the·list·of·numbers:· | | 12 23 45 3 2 1 |  |
| The·number·to·search·for:· | 3 |  | |
| 3·was·found·at·index·3. | | | |

Program No: 7 To write a python program Binary search.

def bsearch(l1,n2,l,r):

mid=(l+r)//2 if l1[mid]==n2:

print("{} was found at index {}.".format(n2,mid)) elif l1[mid]>n2:

bsearch(l1,n2,l,mid-1) else:

bsearch(l1,n2,mid+1,r)

l1=[] n1=int(input("Enter size of list: ")) for i in range(n1):

c1=int(input("Enter your number: "))

l1.append(c1)

l1.sort()

print("After sorting list is: ",l1) n3=int(input("The number to search for: ")) bsearch(l1,n3,0,len(l1))

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. | | | | | |

Program No: 8 WAP to sort elements of a list using Bubble sort.

l=[int(x) for x in input("Enter the list of numbers: ").split()] for i in range(len(l)): for j in range(0,(len(l)-i)-1): if l[j]>l[j+1]:

temp=l[j] l[j]=l[j+1] l[j+1]=temp

print("Sorted list:",l)

|  |  |  |
| --- | --- | --- |
| **Output:** | | |
| Enter·the·list·of·numbers:· | 25 98 74 36 -7 |  |
| Sorted·list:·[-7,·25,·36,·74,·98] | | |

Program No: 9 To write a python program selection sort.

l=[x for x in input("Enter the list of numbers: ").split()] for i in range(len(l)):

max=int(l[i]) temp=i for j in range(i+1,len(l)): if max>int(l[j]):

max=int(l[j])

temp=j

r=l[i] l[i]=l[temp] l[temp]=r

print(l)

|  |  |  |
| --- | --- | --- |
| **Output:** | | |
| Enter·the·list·of·numbers:· | 44 55 2 3 |  |
| ['2',·'3',·'44',·'55'] | | |

Program No: 10 To write a python program Insertion sort.

def ins(l): for i in range(1,len(l)):

temp=l[i] k=int(temp) j=i-1 while j>=0 and k<int(l[j]):

l[j+1]=l[j]

j=j-1

l[j+1]=temp

print(l)

l=[x for x in input("Enter the list of numbers: ").split()] print(l) ins(l)

|  |  |  |
| --- | --- | --- |
| **Output:** | | |
| Enter·the·list·of·numbers:· | 25 98 63 78 99 54 |  |
| ['25',·'98',·'63',·'78',·'99',·'54'] | | |
| ['25',·'54',·'63',·'78',·'98',·'99'] | | |

Program No: 11 Write a program to implement stack using the list

l=[] while True:

a=input("Enter element, 'XXX' to end: ") if a=="XXX":

break

l.append(a)

print("Initial stack") print(l)

print("Elements poped from stack:")

print(l.pop()) print(l.pop()) print(l.pop())

print("Stack after elements are poped:")

print(l)

|  |  |  |  |
| --- | --- | --- | --- |
| **Expected Output:** | | | |
| Enter·element,·'XXX'·to·end:· | a |  | |
| Enter·element,·'XXX'·to·end:· | b |  | |
| Enter·element,·'XXX'·to·end:· | c |  | |
| Enter·element,·'XXX'·to·end:· | d |  | |
| Enter·element,·'XXX'·to·end:· | XXX | |  |
| Initial·stack | | | |
| ['a',·'b',·'c',·'d'] | | | |
| Elements·poped·from·stack: | | | |
| d | | | |
| c | | | |
| b | | | |
| Stack·after·elements·are·poped: | | | |
| ['a'] | | | |

Program No: 12 Write a Python Program to convert infix expression to postfix expression

l=input("Enter infix expression") print("infix expression: ",l)

a="" stck=[] for i in l: if i.isalpha()==True:

a=a+i elifi in "+-\*/()^": if len(stck)==0: stck.append(i) elifi=="(": stck.append(i) elifi==")":

lm=len(stck)-1 while stck[lm]!='(':

lm=lm-1

t=stck.pop()

a=a+t else:

stck.pop() elifi in '+-':

k=len(stck)-1 if stck[k] in '\*/':

r=stck.pop()

a=a+r k=k-1 if len(stck) !=0 and stck[k] in '+-':

r=stck.pop() a=a+r

stck.append(i) else:

stck.append(i) elifstck[k]=='^':

r=stck.pop() a=a+r k=k-1 if len(stck) !=0 and stck[k] in '+-':

r=stck.pop()

a=a+r

stck.append(i) else:

stck.append(i) elifstck[k] in '+-': r=stck.pop()

a=a+r

stck.append(i) else:

stck.append(i) elifi in '\*/':

k=len(stck)-1 if stck[k]=='^':

r=stck.pop()

a=a+r k=k-1 if len(stck)!=0 and stck[k] in '\*/':

r=stck.pop()

a=a+r

stck.append(i) else:

stck.append(i) elifstck[k] in '\*/':

r=stck.pop()

a=a+r

stck.append(i) else:

stck.append(i) elifi in '^':

stck.append(i) if len(stck)==0:

print("postfix expression: ",a) else: while len(stck)!=0:

jj=stck.pop()

a=a+jj

print("postfix expression: ",a)

|  |  |  |
| --- | --- | --- |
| **Output:** | | |
| Enter·infix·expression | a\*b+(c/d) |  |
| infix·expression:··a\*b+(c/d) | | |
| postfix·expression:··ab\*cd/+ | | |

Program No: 13 Aim: Write a Python Program to evaluate postfix expression

f=input("Enter Postfix expression") e=f stck=[] for i in f: if i.isnumeric()==True:

stck.append(i) elifi=='+':

a=int(stck.pop()) b=int(stck.pop())

d=a+b

stck.append(d) elifi=='-':

a=int(stck.pop()) b=int(stck.pop())

d=b-a

stck.append(d) elifi=='\*':

a=int(stck.pop()) b=int(stck.pop())

d=a\*b

stck.append(d) elifi=='/':

a=int(stck.pop()) b=int(stck.pop())

d=a/b

stck.append(d) elifi=='^':

a=int(stck.pop()) b=int(stck.pop())

d=b^a

stck.append(d)

l=stck[0]

print("Result of Postfix expression",e,"is",l)

|  |  |  |
| --- | --- | --- |
| **Output:** | | |
| Enter·Postfix·expression | 231\*+9- |  |
| Result·of·Postfix·expression·231\*+9-·is·-4 | | |

Program No: 14 Aim- Program to implement Merge sort in a non-recursive way.

def mergeS(s): if len(s)>1:

mid=len(s)//2 left=s[:mid] right=s[mid:] mergeS(left) mergeS(right)

i=0 j=0

k=0 while i<len(left) and j<len(right): if left[i]<=right[j]: s[k]=left[i]

i=i+1 k=k+1 else:

s[k]=right[j]

j=j+1

k=k+1 while i<len(left):

s[k]=left[i]

i=i+1

k=k+1 while j<len(right):

s[k]=right[j]

j=j+1 k=k+1

a=int(input("Enter no ofelements"));

s=[] print("enter elements") for i in range(a):

j=int(input()) s.append(j)

print("Given array is ")

print(s) mergeS(s) print("Sorted array is ") print(s)

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Output:** |
| Enter·no·ofelements | | 3 |  |
| enter·elements | | |  |
| 6 |  | |  |
| 4 |  | |  |
| 2 |  | |  |
| Given·array·is· | | |  |
| [6,·4,·2] | | |  |
| Sorted·array·is· | | |  |
| [2,·4,·6] | | |  |

Program no-15 Aim Program to implement Merge sort in a recursive way

def merge(s): if len(s)>1:

mid=len(s)//2 l=s[:mid] r=s[mid:] merge(l) merge(r)

m(s,l,r) def m(s,l,r):

i=0 j=0

k=0 while i<len(l) and j<len(r): if l[i]<=r[j]:

s[k]=l[i] i=i+1 k=k+1 else: s[k]=r[j] j=j+1

k=k+1 while i<len(l):

s[k]=l[i] i=i+1

k=k+1 while j<len(r):

s[k]=r[j] j=j+1

k=k+1

s=[]

n=int(input("Enter no ofelements")) print("enter elements") for i in range(n):

l=int(input()) s.append(l)

merge(s)

print(s)

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | **Output:** |
| Enter·no·ofelements | | 5 |  |
| enter·elements | | |  |
| 9 |  | |  |
| 6 |  | |  |
| 7 |  | |  |
| 8 |  | |  |
| 1 |  | |  |
| [1,·6,·7,·8,·9] | | |  |

Program no-16 Aim Program to implement Quick sort in a recursive way.

def partition(a,l,h):

pivot=a[h] i=l-1 for j in range(l,h): if a[j]<=pivot:

i=i+1

a[i],a[j]=a[j],a[i]

a[i+1],a[h]=a[h],a[i+1] return i+1 def merge(a,l,h): if l<h:

pivot=partition(a,l,h) merge(a,l,pivot-1)

merge(a,pivot+1,h)

a=int(input("Enter no ofelements"))

l=[] print("enter elements") for i in range(a):

s=int(input()) l.append(s)

print("Unsorted Array")

print(l) r=len(l)-1 merge(l,0,r)

print("Sorted Array in Ascending Order:") print(l)

Output:

Enter·no·ofelements4enter·elements

2316

Unsorted·Array

[2,·3,·1,·6]

Sorted·Array·in·Ascending·Order:

[1,·2,·3,·6]

Program no-17 Aim Program to implement binary search using recursion

def binary(l,b):

lb=0

ub=len(l)-1 while lb<=ub: mid=(ub+lb)//2 if l[mid]==b:

print("The element is present at index ",mid)

break elif l[mid]<b:

lb=mid+1 else: ub=mid-1

a=int(input("Enter no of elements")) print("enter elements")

l=[] for i in range(a):

s=int(input()) l.append(s)

b=int(input(" Which element you want to search")) binary(l,b)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | | | | | |

Program no 18 Aim Program to compute factorial using tail recursion def fact(n): if n==0:

return 1 else:

return n\*fact(n-1)

a=int(input("Enter a number: ")) g=fact(a)

print("The factorial of",a,"is",g)

|  |  |  |
| --- | --- | --- |
| **Output:** | | |
| Enter·a·number:· | 5 |  |
| The·factorial·of·5·is·120 | | |

Program no-19 Aim- Program to implement Tower of Hanoi.

def tableofh(n,s,d,a): if n==1:

print("Move disk 1 from source",s,"todestination",d)

return

tableofh(n-1,s,a,d)

print("Move disk",n,"fromsource",s,"todestination",d)

tableofh(n-1,a,d,s)

n=int(input(" enter no of disk")) tableofh(n,'A','B','C')

|  |  |  |
| --- | --- | --- |
| **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 | | |

a

Program to implement the insertion at the front end of the Single Linked list class Node: def \_\_init\_\_(self,data):

self.data=data

self.next=None class linkedlist: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=Node(value) if self.head==None:

self.head=newN else:

newN.next=self.head self.head=newN def disp(self):

n=self.head

print("The Inserted elements at the front end are :") while n!=None: print(n.data) n=n.next

y=1

l=linkedlist() while(y!=3):

y=int(input("Select a Operation: 1.Insertion 2.Display 3.Quit ")) if y==1:

s=int(input("Enter element "))

l.inser(s) elif y==2:

l.disp() elif y==3:

break else:

print("Invalid Option!!!")

Output:

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

|  |
| --- |
| 23 |

Enter·element·

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

|  |
| --- |
| 121 |

Enter·element·

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

|  |
| --- |
| 34 |

Enter·element·

Select·a·Operation:·1.Insertion·2.Display·3.Quit·2The·Inserted·elements·at·the·front·end·are·:

34

121

23

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

b

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

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

self.data=data

self.next=None class ll: def \_\_init\_\_(self):

self.head=None

self.v=-1 def inser(self,value):

newN=node(value) if self.head==None:

self.head=newN self.v=self.v+1 else:

newN.next=self.head self.head=newN self.v=self.v+1 def disp(self):

n=self.head while n!=None: print(n.data) n=n.next def delet(self,x):

temp=self.head if x==0:

temp.next=temp.next.next elif x>self.v:

print("Position is more than number of nodes") else: for i in range(1,x+1): if i==x:

temp.next=temp.next.next

break

Y=1

l=ll() while(Y!=4):

Y=int(input("Select an Operation:\n1.Insert\n2.Deletion\n3.Display\n4.Quit\t")) if Y==1:

s=int(input("Enter Element ")) l.inser(s)

elif Y==2:

z=int(input("Enter a position "))

l.delet(z) elif Y==3:

l.disp() elif Y==4:

break else:

print("Invalid Option!!!")

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |  | | | | |

c

Program to implement traversal in Single Linked List class Node: def \_\_init\_\_(self,data):

self.data=data

self.next=None class ll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=Node(value) if self.head==None:

self.head=newN else:

newN.next=self.head

self.head=newN def display(self):

temp=self.head while temp!= None: print(temp.data) temp=temp.next

l=ll() n=int(input("Enter how many elements would you like to add: ")) for i in range(n):

s=int(input("Enter data elements: ")) l.inser(s)

print("The linked list is: ") l.display()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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:· | | | | |
| 5 | | | | |
| 4 | | | | |
| 3 | | | | |
| 2 | | | | |
| 1 | | | | |

d

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

self.data=data

self.next=None class ll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=node(value) if self.head==None:

self.head=newN else:

newN.next=self.head

self.head=newN def display(self):

temp=self.head while temp!=None: print(temp.data) temp=temp.next def reverse(self):

current=self.head nnext=None prev=None while(current!=None): nnext=current.next current.next=prev prev=current

current=nnext self.head=prev

l=ll() y=1 while(y!=4):

y=int(input("Select a option: 1.Insertion 2.Reversal 3.Display 4.Quit ")) if y==1:

s=int(input("Enter number ")) l.inser(s) elif y==2:

l.reverse() elif y==3:

l.display()

elif y==4:

break

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

|  |
| --- |
| 15 |

Enter·number·

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

e

Program to implement searching in Single Linked List

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

self.data=data

self.next=None class ll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=node(value) if self.head==None:

self.head=newN else:

newN.next=self.head

self.head=newN def search(self,v):

temp=self.head

c=0 while temp!=None: if temp.data==v:

c=c+1 break

temp=temp.next if c==1: print("Item found") else:

print("item not found")

def display(self):

temp=self.head while temp!=None: print(temp.data)

temp=temp.next l=ll()

y=1 while y!=4:

y=int(input("Select Operation:\n1.Insertion\n2.Searching\n3.Display\n4.Quit\t")) if y==1:

s=int(input("Enter elements "))

l.inser(s)

elif y==2:

j=int(input("Enter a key to search ")) l.search(j)

elif y==3:

l.display() elif y==4:

break

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·foundSelect·Operation:

1.Insertion

2.Searching

3.Display 4.Quit→3

9

7

Select·Operation:

1.Insertion

2.Searching

3.Display

4.Quit→4

f

Program to implement Updation in single linked list

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

self.data=data

self.next=None class ll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=node(value) if self.head==None:

self.head=newN else:

newN.next=self.head

self.head=newN def display(self):

temp=self.head while temp!=None: print(temp.data)

temp=temp.next def update(self,pos,val):

temp=self.head if pos==0:

temp.data=val else:

temp=temp.next for i in range(1,pos+1): if i==pos: temp.data=val

break

temp=temp.next l=ll()

y=1 while y!=4:

y=int(input("Select Operation\n1.Insertion\n2.Updation\n3.Display\n4.Quit\t")) if y==1:

s=int(input("Enter element "))

l.inser(s) elif y==2:

j=int(input("Enter the index to update ")) k=int(input("Enter a value to update "))

l.update(j,k) elif y==3:

l.display()

elif y==4: break

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |  | | | | | |

g

Program to implement Sorting in Single linked list

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

self.data=data

self.next=None class ll: def \_\_init\_\_(self):

self.head=None def inse(self,value):

newN=Node(value) if self.head==None:

self.head=newN else:

temp=self.head while temp.next != None:

temp=temp.next

temp.next=newN def sort(self): if self.head==None: print(" List is empty")

return else:

current=self.head index=None while current!=None: index=current.next while index != None:

if current.data>index.data: temp=current.data

current.data=index.dataindex.data=temp index=index.next current=current.next def display(self):

temp=self.head while temp!=None: print(temp.data) temp=temp.next

l=ll()

y=1

while(y!=4):

y=int(input("Select a Operation 1.Insertion 2.Sorting 3.Display 4.Quit ")) if y==1:

s=int(input("Enter a element "))

l.inse(s) elif y==2:

l.sort() elif y==3:

l.display() elif y==4:

break

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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· | | | 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 |  |

a

Program to implement Insertion operation by adding the elements at the end f the doublyLinked List class Node: def \_\_init\_\_(self,data):

self.data=data self.next=None self.prev=None class dll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

new=Node(value) if self.head==None:

self.head=new else:

temp=self.head while temp.next!= None:

temp=temp.next temp.next=new new.prev=temp def display(self):

temp=self.head while temp!= None: print(temp.data)

temp=temp.next l=dll() y=1 while y!=3:

y=int(input("Select Opertion\n1.Insertion\n2.Display\n3.Quit\t")) if y==1:

s=int(input("enter element "))

l.inser(s) elif y==2:

print("Adding a node to the end of the list: ")

l.display() elif y==3:

break

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  | | |

c

Program to implement double linked list traversal in forward direction and traversal in thereverse direction in double linked list

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

self.data=data self.prev=None self.next=None class dll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=Node(value) if self.head==None:

self.head=newN else:

newN.next=self.head self.head.prev=newN

self.head=newN def forward(self):

temp=self.head

print("Traversal in forward direction") while temp!=None: print(temp.data) temp=temp.next def backward(self):

temp=self.head

print("Traversal in reverse direction") while temp.next!=None:

temp=temp.next while temp!=None: print(temp.data)

temp=temp.prev l=dll() y=int(input("Enter Number of Elements to Insert in DoublyLinkedList ")) for i in range(y):

x=int(input("Enter Element ")) l.inser(x)

l.forward()

l.backward()

**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 | | | | |

d

Program to implement searching in Double Linked List

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

self.data=data self.prev=None self.next=None class dll: def \_\_init\_\_(self):

self.head=None def inser(self,value):

newN=Node(value) if self.head==None:

self.head=newN else:

temp=self.head while temp.next!=None: temp=temp.next temp.next=newN newN.prev=temp def search(self,v):

temp=self.head

c=0 s=0 while temp!=None:

s=s+1 if temp.data==v:

c=c+1

break

temp=temp.next if c==1:

print("Node is present in the list at the position :",s) else:

print("Node is not present in the list") def display(self):

temp=self.head while temp!=None: print(temp.data) temp=temp.next

l=dll() y=1 while y!=4: y=int(input("Select a operation:\n1.Insertion\n2.Searching\n3.Display\n4.Quit\t")) if y==1:

x=int(input("Enter Element "))

l.inser(x) elif y==2:

z=int(input("Enter Element to Search "))

l.search(z) elif y==3:

l.display() elif y==4:

break

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