Datastructures Lab

[**Week 1**](#_heading=h.dqi8ntjvoyvj) **2**

[Task-1](#_heading=h.rd3j1l9rb2ds) 2

[Task-2](#_heading=h.9hiv8bz3nfs0) 4

[Task-3](#_heading=h.9uluaqre1qjl) 6

[**WEEK 2**](#_heading=h.xdtwds9c0on6) **8**

[Task-1](#_heading=h.ku1vsyc61hcd) 9

[Task-2](#_heading=h.fsijfwqe7n7j) 10

[Task-3](#_heading=h.wi0yl4bcvosv) 12

[**WEEK 3**](#_heading=h.z6svebnkixnu) **14**

[Task-1](#_heading=h.9vsvvzl1wjhs) 14

[Task-2](#_heading=h.eqtsp9b8uv26) 15

[Task-3](#_heading=h.4k5ibssh4zgt) 16

[**WEEK 4**](#_heading=h.j217jb9fncb8) **18**

[Task-1](#_heading=h.h0c6epi6az43) 18

[Task-2](#_heading=h.hqxhb62lnrpm) 20

[**WEEK 5**](#_heading=h.x1nioqt1xyj3) **21**

[Task-1](#_heading=h.dfv2v3xpdy5i) 21

[Task-2](#_heading=h.39c124ena3ba) 24

[Task-3](#_heading=h.gbrh6q4px79y) 27

[**WEEK 6**](#_heading=h.rizkm4oq2tok) **31**

[Task-1](#_heading=h.cn0xf511utlm) 31

[Task-2](#_heading=h.w718oj7gccg1) 33

[**WEEK 7**](#_heading=h.9zqrsn45gly2) **37**

[Task-1](#_heading=h.nw0kvx5i8qjj) 37

[TASK-2](#_heading=h.cx2548j3jgul) 38

[**WEEK-8**](#_heading=h.7iprl438ir5p) **42**

[Task-1](#_heading=h.yzoaavvf0p1i) 42

[Task-2](#_heading=h.zdj2eyxvengm) 45

[Task-3](#_heading=h.21r311h59xpp) 48

[Task-4](#_heading=h.689loools8pt) 52

[**WEEK-9**](#_heading=h.bkennk8c5tgn) **56**

[Task-1](#_heading=h.73j6imzicrlw) 56

[Task-2](#_heading=h.jdzz4vn45wc7) 60

[Task-3](#_heading=h.52l8ihgt6il) 60

[Task-4](#_heading=h.gmg55x4s6rsl) 61

[**WEEK 10:**](#_heading=h.4ugztf2tw4a) **61**

[Task-1](#_heading=h.lvg3hjqz8do0) 62

[Task-2](#_heading=h.otzqsckojrpb) 64

[Task-3](#_heading=h.b20hvd6idxfp) 64

[Task-4](#_heading=h.2qucm49c0ib8) 65

[**WEEK 11:**](#_heading=h.gqc3uh6x8lhz) **65**

[Task-1](#_heading=h.xqvlew1jthaw) 65

[Task-2](#_heading=h.v0wui31h42l9) 68

[**WEEK 12:**](#_heading=h.b0lxvnhdo5cj) **71**

[Task-1](#_heading=h.fnmp4zcx8fi1) 71

[Task-2](#_heading=h.e1daau1hfjyl) 71

# Week 1

**Datastructures Lab Task II/IV B.Tech**

Batch-1: 09-November-2021 @ 9.30 to 12

Batch-2: 11-November-2021 @ 8.40 to 11.10

## Task-1

We have discussed linear search algorithms in the class. Implement the same algorithm using any of the languages like C/Python. Write all the possible test cases. Write down your observations.

Sample input

Enter number of elements 8

Enter 8 elements

44,16,18,164,47,10,0,-68

Enter element to search 10

Sample output

10 is present at location 5

**Program(C):**

#include<stdio.h>

int main(){

int a[100],i,found=0,n,key;

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter %d elements:",n);

for(i=0;i<=n-1;i++)

scanf("%d",&a[i]);

printf("Enter element to search:");

scanf("%d",&key);

for(i=0;i<=n-1;i++)

if(a[i]==key){

printf("%d is present at location %d",key,i);

found=1;

break;

}

if(found==0)

printf("Element is not found");

return 0;

}

or

**Program(Python):**

n=int(input("enter the size of array:"))

print("enter the elements:")

a=list(map(int,input().split()))

key=int(input("enter the element to be found:"))

f=0

for i in range(n):

if a[i]==key:

print( "{} is the location of {}".format(i,key))

f=1

break

if f==0:

print( "element not found")

**Output 1:**

Enter number of elements:5

Enter 5 elements:1 2 -5 6 0

Enter element to search:-5

-5 is present at location 2

**Output 2:**

Enter number of elements:5

Enter 5 elements: 1 2 3 4 5

Enter element to search:6

Element is not found

## Task-2

Suppose you are implementing a Linear search algorithm using any of the languages like C/Python. In the given array if there is a possibility of multiple occurrences of some elements. In such a case how to identify the location of the element. Design and implement the solution for the same.

Sample input

Enter number of elements 6

Enter 6 elements

44,16,18,16,47,16

Enter element to search 16

Sample output

16 is present at location 1

16 is present at location 3

16 is present at location 5

**Program(C):**

#include<stdio.h>

int main(){

int a[100],i,found=0,n,key;

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter %d elements",n);

for(i=0;i<=n-1;i++)

scanf("%d",&a[i]);

printf("Enter element to search:");

scanf("%d",&key);

for(i=0;i<=n-1;i++)

if(a[i]==key){

printf("%d is present at location %d\n",key,i);

found=1;

}

if(found==0)

printf("Element is not found");

return 0;

}

**Program(Python):**

n=int(input("enter the size of array:"))

print("enter the elements:")

a=list(map(int,input().split()))

key=int(input("enter the element to be found:"))

f=0

for i in range(n):

if a[i]==key:

print( "{} is at location of {}".format(key,i))

f=1

if f==0:

print( "element not found")

**Output 1:**

Enter number of elements:5

Enter 5 elements 1 2 3 4 2 2

Enter element to search:2

2 is present at location 1

2 is present at location 4

**Output 2:**

Enter number of elements:6

Enter 6 elements 1 54 3 32 5 -5

Enter element to search:4

Element is not found

## Task-3

Rithick gets a lottery ticket and checks each number in the list to see whether he has won the lottery or not. Since there are many numbers,he finds it tedious to check each ticket number manually. So he decides to write a code to check whether he has won the lottery or not. Help Rithick write a code to find his lottery ticket number from the given ticket numbers.

**Input Format:**

First line of the input consists of n, that corresponds to the total number of lottery tickets.

Next n lines consist of Integers, that corresponds to the given lottery ticket numbers.

Last line consists of an Integer 'l', which corresponds to Rithick's lottery ticket number.

**Output Format:**

Output consists of the string "Congratulations! You have won the lottery" or "Sorry your ticket number is not there. Better luck next time", according to the search result.

**Sample Input and Output:**  
**[All text in bold corresponds to input and the rest corresponds to output]**  
Enter the total number of tickets:  
**5**  
Enter the ticket number 1:  
**4521**  
Enter the ticket number 2:  
**3589**  
Enter the ticket number 3:  
**147852**  
Enter the ticket number 4:  
**2365**  
Enter the ticket number 5:  
**8965**  
The ticket numbers are:  
4521 3589 147852 2365 8965  
Enter the ticket number to be searched:  
**8965**  
The ticket number 8965 is found at position 5  
Congratulations!You have won the lottery

**Program(C):**

#include<stdio.h>

int main(){

int a[100],i,found=0,n,I;

printf("Enter the total number of tickets:");

scanf("%d",&n);

printf("Enter %d elements\n",n);

for(i=0;i<=n-1;i++){

printf("Enter the ticket number %d:",i+1);

scanf("%d",&a[i]);

}

printf("Enter the ticket number to be searched:");

scanf("%d",&I);

for(i=0;i<=n-1;i++)

if(a[i]==I){

printf("The ticket number %d is found at position %d",I,i);

printf("Congratulations! You have won the lottery");

found=1;

break;

}

if(found==0)

printf("Sorry! You Lost");

return 0;

}

**Program(Python):**

n=int(input("enter the number of tickets:"))

print("entre the tickets numbers:")

a=list(map(int,input().split()))

i=int(input("enter the ticket number to be found:"))

f=0

for j in range(n):

if a[j]==i:

print( "Congratulations! You have won the

lottery")

f=1

if f==0:

print( "Sorry your ticket number is not there. Better

luck next time")

**Output 1:**

Enter the total number of tickets:5

Enter 5 elements

Enter the ticket number 1:1234

Enter the ticket number 2:1376

Enter the ticket number 3:2456

Enter the ticket number 4:2967

Enter the ticket number 5:5674

Enter the ticket number to be searched:2967

The ticket number 2967 is found at position 3

Congratulations! You have won the lottery

**Output 2:**

Enter the total number of tickets:4

Enter 4 elements

Enter the ticket number 1:1234

Enter the ticket number 2:4566

Enter the ticket number 3:8976

Enter the ticket number 4:3457

Enter the ticket number to be searched:6757

Sorry! You Lost

# 

# WEEK 2

**Data structures Lab Task II/IV B.Tech**

Batch-1: 16-November-2021 @ 9.30 to 12

Batch-2: 18-November-2021 @ 8.40 to 11.10

## Task-1

We have discussed the Binary search algorithm in the class. Implement the same algorithm using any of the languages like C/Python. Write all the possible test cases. Write down your observations.

Sample input

Enter number of elements 8

Enter 8 elements

44,16,18,164,47,10,0,-68

Enter element to search 10

Sample output

10 is present at location 5

**Program(Python):**

n=int(input("Enter number of elements:"))

print(f"Enter {n} elements")

l=list(map(int,input("enter elements\n").split(",")))

key=int(input("Enter element to search:"))

low,high,found=0,n-1,0

while(low<=high):

mid=(low+high)//2

if(key>l[mid]):

low=mid+1

elif(key<l[mid]):

high=mid-1

else:

print(f"{key} is present at location {mid}")

found=1

break

if(found==0):

print(f"{key} is not found")

**Output** 1**:**

Enter number of elements:6

Enter 6 elements

12,34,67,89,121,245

Enter element to search:121

121 is present at location 4

**Output 2:**

Enter number of elements:5

Enter 5 elements

12,34,56,78,90

Enter element to search:45

45 is not found

## Task-2

Once there lived a king and a queen. They were very rich and led a happy life.

The King's brother Humayun was jealous of him and he wanted to become the King. So in order to become the King, Humayun carries the Queen away and keeps her in a jail which are numbered in sorted sequence.

Humayun informs the King that he took away the Queen and if the King agrees him to become the King, he would leave the Queen. The King has a special power of flying. He flies and reaches the place where Humayun has hidden the Queen and Humayun has given clue to King about the jail number of the Queen. Luckily the King finds the key of the jail where the Queen is locked.But Humayun might have tried to fool the King by saying wrong jail number of Queen. The King already knew how many jails were there. He reaches the centre jail and checks whether the jail number and the key number are equal. He also carries a small machine with him to find the next jail to be visited by him. The machine initially contains the low(low=1) and the high value(high=total number of jails), and calculates the mid value (the jail which is to be visited). If the mid value and key value are equal, the King opens the lock and takes the Queen away with him. If the key number is greater than the mid value, the King increases the mid value by 1 in the machine and assigns it to the low value. If the key number is less than the mid value, the King decreases the mid value by 1 in the machine and assigns it to the high value. The King does this until he finds that the key value and the mid value to be equal.

**Sample Input and Output1:**

Enter the total number of jails in Humayun's Place:

**5**

Enter the jail number 1

**6**

Enter the jail number 2

**89**

Enter the jail number 3

**91**

Enter the jail number 4

**105**

Enter the jail number 5

**200**

Enter the clue given by Humayun:

**105**

Hurray!The King rescued the Queen

**Sample Input and Output 2:**

Enter the total number of jails in Humayun's Place:

**6**

Enter the jail number 1

**56**

Enter the jail number 2

**68**

Enter the jail number 3

**156**

Enter the jail number 4

**196**

Enter the jail number 5

**204**

Enter the jail number 6

**895**

Enter the clue given by Humayun:

**4**

The King has been fooled by Humayun

**Program(Python):**

n=int(input("Enter the total number of jails in Humayun's Place:"))

l=[]

for i in range(n):

i=int(input(f"Enter the jail number {i+1}:"))

l.append(i)

l.sort()

key=int(input("Enter the clue given by Humayun:"))

low,high,found=0,n-1,0

while(low<=high):

mid=(low+high)//2

if(key>l[mid]):

low=mid+1

elif(key<l[mid]):

high=mid-1

else:

print("Hurray!The King rescued the Queen")

found=1

break

if(found==0):

print("The King has been fooled by Humayun")

**Output 1:**

Enter the total number of jails in Humayun's Place:5

Enter the jail number 1:234

Enter the jail number 2:5634

Enter the jail number 3:3245

Enter the jail number 4:56

Enter the jail number 5:34

Enter the clue given by Humayun:43

The King has been fooled by Humayun

**Output 2:**

Enter the total number of jails in Humayun's Place:5

Enter the jail number 1:123

Enter the jail number 2:234

Enter the jail number 3:345

Enter the jail number 4:456

Enter the jail number 5:678

Enter the clue given by Humayun:456

Hurray!The King rescued the Queen

## Task-3

Given an Array A of N of integers.You have to find number of subarrays having Primicity less than or equal to K

Primicity of a sub-array is defined as a number of primes in that subarray.

**Input:**

First line contains N and K

Second line contains N integers.

**Output:**

Print the Number of sub-arrays having Primicity<= K

**Program(Python):**

def primicity(a):

count=0

NumberOfPrime=0

for i in a:

for j in range(2,(i//2)+1):

if (i % j) == 0:

break

else:

NumberOfPrime+=1

if(NumberOfPrime<=k):

count+=1

return count

n=int(input("Enter number of elements"))

l=list(map(int,input("enter elements:").split("," )))

k=int(input("Enter K:"))

a=[]

for i in range(len(l)+1):

for j in range(i+1,len(l)+1):

sub=l[i:j]

a.append(sub)

sum=0

for i in a:

sum+=primicity(i)

print(sum)

**Output 1:**

Enter number of elements:4

enter elements:1,2,3,4

Enter K:2

8

**Output 2:**

Enter number of elements 8

enter elements:1,2,3,4,5,6,78 ,8

Enter K:5

36

# WEEK 3

**Datastructures Lab Task II/IV B.Tech**

Batch-1: 23-November-2021 @ 9.30 to 12

Batch-2: 25-November-2021 @ 8.40 to 11.10

## Task-1

In a class room the teacher decides to shuffle all the students so that each student collaborates with other students too and she also checks that board is visible to all students. In a bench n number of students can be seated. She calls the students and makes them stand in a line.

She checks first two person and if the first person is taller than the second person he is put in second place and second in first, i.e., their positions are swapped. After swapping, the teacher checks second and third person height and if the second person is taller than third person their positions are swapped. She repeats this till the end. She finds that the last student is the tallest student after completing one iteration.

So she repeats the process for (n-1) times so that all students are sorted. She stops the process in the middle as soon as she find that the students are sorted.

For ex:

Consider 5 number of students can be seated in a bench

Their height order is in

6 4 3 2 5

During first comparison, adjacent numbers 6 and 4 is compared, 6 is greater than 4 hence the order becomes 4 6 3 2 5.

Again the teacher checks 6 and 3. 6 is greater than 3 hence the numbers are swapped and the order becomes 4 3 6 2 5.

She compares 6 and 2 and they get swapped and the order becomes 4 3 2 6 5. She compares 6 and 5 they get swapped and the order becomes 4 3 2 5 6.

She sees that they are not in order she again takes adjacent person heights and she swaps. She keeps doing this until the order becomes 2 3 4 5 6.

Help the teacher write a program to do bubble sorting inorder to sort the height in order.

**For ex:**

Consider 5 number of students can be seated in a bench

Their height is in order

6 4 3 2 5

During first comparison, adjacent numbers 6 and 4 is compared, 6 is greater than 4 hence the order becomes **4 6 3 2 5**.

Again the teacher checks 6 and 3, 6 is greater than 3. Hence the numbers are swapped and the order becomes **4 3 6 2 5**.

She compares 6 and 2 and they get swapped and the order becomes **4 3 2 6 5**. She compares 6 and 5, they get swapped and the order becomes **4 3 2 5 6**.

She sees that they are not in order. So she again compares adjacent person heights and she swaps. She repeats this until the heights get sorted. Finally the students will be seated in this order - **2 3 4 5 6**.

Help the teacher write a program to do this task.

**Input Format:**Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next n integers correspond to the elements in the array.

**Output Format:**Refer sample output for formatting specs.

**Program(Python):**

#Bubble Sort

l=list(map(int,input("Enter the number of students and their heights:").split()))

n=l[0]

for i in range(n-1):

for j in range(1,n):

if l[j]>l[j+1]:

l[j],l[j+1]=l[j+1],l[j]

print(f"Finally the students will be seated in this order -{l[1:n+1]}")

**Output 1:**

Enter the number of students and their heights:5 24 1 2 4 3

Finally the students will be seated in this order - [1, 2, 3, 4, 24]

**Output 2:**

Enter the number of students and their heights:8 12 36 28 39 47 57 281 1

Finally the students will be seated in this order -[1, 12, 28, 36, 39, 47, 57, 281]

## Task-2

Two friends Reema and Rita decide to play cards. But Reema doesn't know to play the cards. So Rita decides to teach her how to play cards. Rita distributes 13 cards to each other which are in unsorted order and teaches her how to make set with the cards without considering the symbols.

Rita first takes two cards and checks which card has the lesser number and the card with the least number is put first and larger at second.

She takes third card and checks it with second card. If the third card number is less than the second card she inserts the third card at the second position and checks newly inserted second card with the first card. If the newly inserted second card number is less than the first card, she inserts the second card at the first place and this goes on till all the cards get sorted.

Rita finally learns how to order the cards.

Write a program to perform insertion sort on an array of n elements.

**Input Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next n integers correspond to the elements in the array.

**Output Format:**

Refer sample output for formatting specs

**Program(Python):**

#Insertion Sort

a=list(map(int,input("Enter the number of cards and their numbers:").split()))

n=a[0]

a.pop(0)

for step in range(1, len(a)):

key = a[step]

j = step - 1

while j >= 0 and key < a[j]:

a[j + 1] = a[j]

j = j - 1

a[j + 1] = key

print(f"Sorted order - {a}")

**Output 1:**

Enter the number of cards and their numbers:5 1 3 2 5 4

Sorted order - [1, 2, 3, 4, 5]

**Output 2:**

Enter the number of cards and their numbers:8 21 45 21 12 34 3 4 0

Sorted order - [0, 3, 4, 12, 21, 21, 34, 45]

## Task-3

Its the first day for the students at school and the students enter the class and get seated at random places without any height order. So the students who are short and sitting back are not able to see the board since they sit behind taller students.

Understanding this difficulty, the teacher decides to make the students sit in height order.

Suppose there are n students in the class. She makes all the students to stand in a line and compares the first student's height with the remaining (n-1) students. If the first student's height is greater than the ith student, then the taller person goes to the ith place and ith student comes to the first place. Again the new first student's height is compared with remaining students and if his height is greater than ith student the first student goes to ith place and ith place student comes to first place and this goes on till the end.

This process continues for all the students. Finally the students are in height order.

Write a program to perform selection sort on an array of n elements.

**Input Format:**Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array.The next n integers correspond to the elements in the array.

**Output Format:**Refer sample output for formatting specs.

**Program(Python):**

#Selection sort

l=list(map(int,input("Enter the number of students and their heights:").split()))

n=l[0]

l.pop(0)

for i in range(n-1):

for j in range(i,n):

if l[j]<l[i]:

l[j],l[i]=l[i],l[j]

print(l)

**Output 1:**

Enter the number of students and their heights:5 1 1 2 3 4

[1, 1, 2, 3, 4]

**Output 2:**

Enter the number of students and their heights:5 32 22 11 21 32

[11, 21, 22, 32, 32]

# WEEK 4

## Task-1

We have discussed sorting algorithms in the class. Implement the same algorithm using any of the languages like C/Python. Given a group of unordered elements. Design an algorithm/pseudocode based on divide and conquer to make the elements into sorted lists. Write all your observations. Elaborate the test cases.

**Note: Use the Logic discussed in the class**

**Algorithm as below**

**MergeSort(arr[], l, r)**

If r > l

**1.** Find the middle point to divide the array into two halves:

middle m = l+ (r-l)/2

**2.** Call mergeSort for first half:

Call mergeSort(arr, l, m)

**3.** Call mergeSort for second half:

Call mergeSort(arr, m+1, r)

**4.** Merge the two halves sorted in step 2 and 3:

Call merge(arr, l, m, r)

**Test case-1**

Enter number of Elements

12

Apply the process of merge sort and sort the following list of elements:

Mar, May, Nov, Aug, Apr, Jan, Dec, Jul, Feb, Jun, Oct, Sep

**Test case-2**

Enter number of Elements

6

Enter array of elements

12,45,10,33,55,50

Sorted array is

10,12,33,45,50,55

**Program(Python):**

#Merge Sort

def merge(l,r,a):

i,j,k=0,0,0

while(i<len(l) and j<len(r)):

if(l[i]<=r[j]):

a[k]=l[i]

i+=1

else:

a[k]=r[j]

j+=1

k+=1

while(i<len(l)):

a[k]=l[i]

i+=1

k+=1

while(j<len(r)):

a[k]=r[j]

j+=1

k+=1

def mergesort(a):

if len(a)<2:

return 0

mid=len(a)//2

l=a[:mid]

r=a[mid:]

mergesort(l)

mergesort(r)

merge(l,r,a)

return a

n=int(input("Enter number of Elements"))

a= list(map(int,input("Enter the array of elements:").split()))[:n]

mergesort(a)

print(a)

**Output 1:**

Enter number of Elements5

Enter the array of elements:1 4 3 2 5

[1, 2, 3, 4, 5]

**Output 2:**

Enter number of Elements8

Enter the array of elements:1 4 6 8 2 3 5 7

[1, 2, 3, 4, 5, 6, 7, 8]

## Task-2

We have discussed sorting a algorithm in the class. Implement the same algorithm using any of the languages like C/Python. Follow the following LOC.

Sample Input and Output

Enter number of Elements

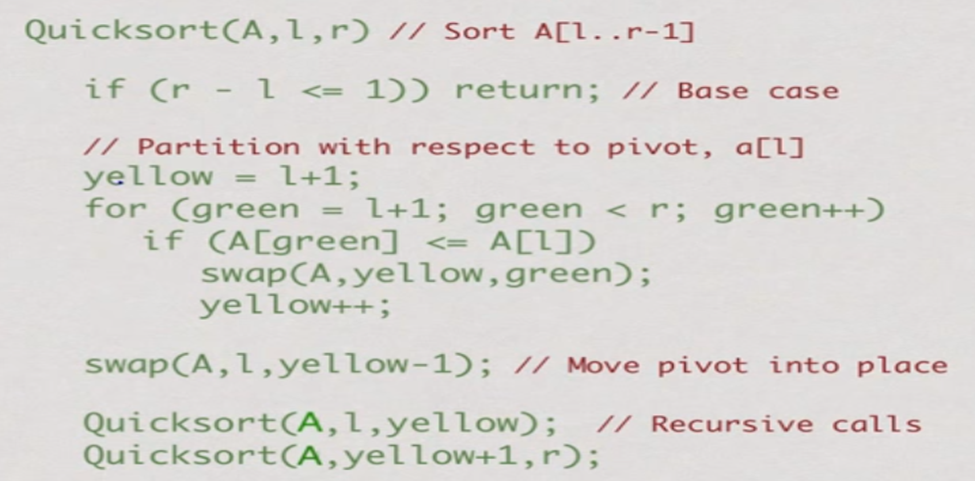
6

Enter array of elements

12,45,10,33,55,50

Sorted array is

10,12,33,45,50,55



**Program(Python):**

def partition(a,l,r):

pivot=a[r]

i=l-1

for j in range(l,r+1):

if(a[j]<pivot):

i+=1

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

a[i+1],a[j]=a[j],a[i+1]

return i+1

def quicksort(a,l,r):

if(l<r):

pi=partition(a,l,r)

quicksort(a,l,pi-1)

quicksort(a,pi+1,r)

return a

n=int(input("Enter number of Elements "))

a= list(map(int,input("Enter the array of elements:").split()))[:n]

print(quicksort(a,0,len(a)-1))

**Output 1:**

Enter number of Elements 8

Enter the array of elements:1 4 2 6 3 8 11 7

[1, 2, 3, 4, 6, 7, 8, 11]

**Output 2:**

Enter number of Elements 5

Enter the array of elements:1 3 2 1 1

[1, 1, 1, 2, 3]

# WEEK 5

**Data structures Lab Task II/IV B.Tech**

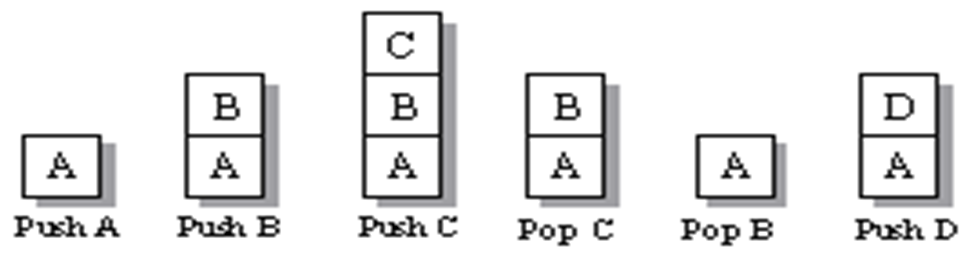
Batch-1: 07-December-2021 @ 9.30 to 12

Batch-2: 09-December-2021 @ 8.40 to 11.10

## Task-1

We have discussed stack operations called push () and pop () in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to stack (2) Delete element from stack (3) Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases.

Sample test case is as below:



**Program(C):**

#include <stdio.h>

#include<stdlib.h>

int tos=0;

int max\_size=6;

int a[100];

void push(int y){

if(tos>=max\_size)

printf("Stack Overflow\n");

else{

a[tos]=y;

tos++;

}

}

void pop(){

if(tos<=0)

printf("Stack Empty\n");

else{

printf("Element Popped is %d\n",a[tos-1]);

a[tos-1]=0;

tos--;

}

}

int display(){

if(tos==0)

printf("Stack is Empty\n");

else{

printf("Elements of stack are:\n");

for(int i=0;i<=tos-1;i++)

printf("%d ",a[i]);

printf("\n");

}}

int main(){

int m,a;

while(1){

printf("MENU\n1:Push 2:Pop 3:Display 4:Exit\n");

scanf("%d",&m);

switch(m){

case 1:

printf("Enter a number to push:\n");

scanf("%d",&a);

push(a);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

printf("Exiting Program...");

exit(0);

break;

}}

return 0;

}

**Output 1:**

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

2

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Elements of stack are:

2

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

4

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Elements of stack are:

2 4

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Element Popped is 4

MENU

1:Push 2:Pop 3:Display 4:Exit

4 3

Elements of stack are:

2

MENU

1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

**Output 2:**

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Stack Empty

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

5

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Element Popped is 5

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Elements of stack are:

Stack is Empty

MENU

1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program…

## Task-2

Design and implement C program to check a string is palindrome or not using a stack. A **stack** is LAST IN FIRST OUT (LIFO) data structure. The element which is inserted last, is accessed first. Insertion and deletion of elements happens only at top of the **Stack**. The sequence of exit of elements from a stack is reverse of the sequence of their entry in stack.

Sequence of Entry.

A --> B --> C -- > D --> E

Sequence of Exit.

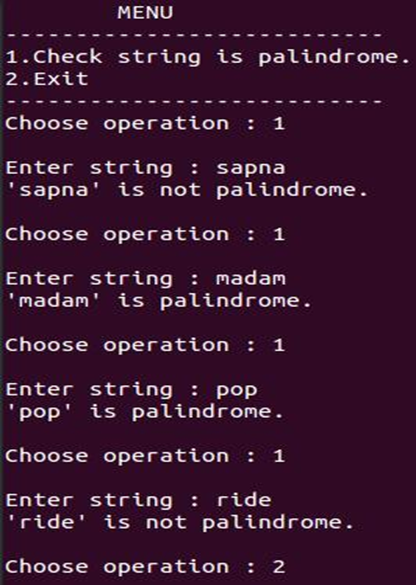
E --> D --> C --> B --> A

**Algorithm to check palindrome string using stack**

* Find the length of the input string using strlen function and store it in a integer variable "length".
* Using a for loop, traverse input string from index 0 to length-1 and push all characters in stack.
* Remove (Pop) characters from stack one by one using a for loop and compare it with corresponding character of input string from beginning(traverse from index 0 to length-1). If we found a mismatch the input string is not a palindrome string otherwise **palindrome string**.

Write all possible examples supported by relevant test cases.

Sample test case is as below:



**Program(C):**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int tos = -1;

char arr[50] ;

void push(char x){

tos++;

arr[tos] = x;

}

int pop(){

char y = arr[tos];

tos--;

return y;

}

int main(){

int check,key;

char word[50];

while(1){

printf("1.Check string is palindrome\n");

printf("2.exit\n");

printf("Choose Operation : ");

scanf("%d",&key);

switch(key){

case 1:

printf("Enter string : ");

scanf("%s",word);

int length = strlen(word);

for(int i=0;i<length;i++){

push(word[i]);

}

for(int i=0;i<length;i++){

char a = pop();

if (a != word[i]){

check = 0;

break;

}

}

if (check==0){

printf("%s is not a palindrome\n",word);

}

else{

printf("%s is a palindrome\n",word);

}

break;

case 2:

exit(0);

}

}

return 0;

}

**Output 1:**

1.Check string is palindrome

2.exit

Choose Operation : 1

Enter string : mam

mam is a palindrome

1.Check string is palindrome

2.exit

Choose Operation : 1

Enter string : ram

ram is not a palindrome

1.Check string is palindrome

2.exit

Choose Operation : 2

**Output 2:**

1.Check string is palindrome

2.exit

Choose Operation : 1

Enter string : aa

aa is a palindrome

1.Check string is palindrome

2.exit

Choose Operation : 1

Enter string : aba

aba is a palindrome

1.Check string is palindrome

2.exit

Choose Operation : 2

## Task-3

You have an empty sequence, and you will be given

queries. Each query is one of these three types:

1 x -Push the element x into the stack.

2 -Delete the element present at the top of the stack.

3 -Print the maximum element in the stack.

**Function Description**

Complete the *getMax* function in the editor below.

*getMax* has the following parameters:

- *string operations[n]:* operations as strings

**Returns**

- *int[]:* the answers to each type 3 query

**Input Format**

The first line of input contains an integer, The next lines each contain an above mentioned query.

**Constraints**

All queries are valid.

**Sample Input**

STDIN Function

----- --------

10 operations[] size n = 10

1 97 operations = ['1 97', '2', '1 20', ....]

2

1 20

2

1 26

1 20

2

3

1 91

3

**Sample Output**

26

91

**Program(C):**

#include <stdio.h>

#include<stdlib.h>

int tos=0;

int max\_size=6;

int a[100];

void push(int y){

if(tos>=max\_size)

printf("Stack Overflow\n");

else{

a[tos]=y;

tos++;

}

}

void pop(){

if(tos<=0)

printf("Stack Empty\n");

else{

printf("Element Popped is %d\n",a[tos-1]);

a[tos-1]=0;

tos--;

}

}

int getMax(){

int max=0;

if(tos==0)

printf("Stack is Empty\n");

else{

printf("Elements of stack are:\n");

for(int i=0;i<=tos-1;i++)

if(a[i]>max)

max=a[i];

printf("%d ",max);

}}

int main(){

int m,a;

while(1){

printf("MENU\n1:Push 2:Pop 3:Max Element 4:Exit\n");

scanf("%d",&m);

switch(m){

case 1:

printf("Enter a number to push:\n");

scanf("%d",&a);

push(a);

break;

case 2:

pop();

break;

case 3:

getMax();

break;

case 4:

printf("Exiting Program...");

exit(0);

break;

}}

return 0;

}

**Output 1:**

MENU

1:Push 2:Pop 3:Max Element 4:Exit

1

Enter a number to push:

11

MENU

1:Push 2:Pop 3:Max Element 4:Exit

1

Enter a number to push:

16

MENU

1:Push 2:Pop 3:Max Element 4:Exit

3

Max Element:

16

MENU

1:Push 2:Pop 3:Max Element 4:Exit

2

Element Popped is 16

MENU

1:Push 2:Pop 3:Max Element 4:Exit

3

Max Element:

11

MENU

1:Push 2:Pop 3:Max Element 4:Exit

4

Exiting Program...

**Output 2:**

MENU

1:Push 2:Pop 3:Max Element 4:Exit

1

Enter a number to push:

12

MENU

1:Push 2:Pop 3:Max Element 4:Exit

2

Element Popped is 12

MENU

1:Push 2:Pop 3:Max Element 4:Exit

3

Stack is Empty

MENU

1:Push 2:Pop 3:Max Element 4:Exit

1

Enter a number to push:

12

MENU

1:Push 2:Pop 3:Max Element 4:Exit

2

Element Popped is 12

MENU

1:Push 2:Pop 3:Max Element 4:Exit

2

Stack Empty

MENU

1:Push 2:Pop 3:Max Element 4:Exit

4

Exiting Program...

# WEEK 6

**Datastructures Lab Task II/IV B.Tech**

Batch-1: 14-December-2021 @ 9.30 to 12

Batch-2: 16-December-2021 @ 8.40 to 11.10

## Task-1

In a class room we have discussed the stack operations called push() and pop(). Based on these operations, design and implement C/Python code to evaluate postfix expressions. Hint: Use the following statement in the push function **stack[top]=(int)(post[tmp]-48);** for ASCII value conversion from string to numbers. For examples please refer<http://www.btechsmartclass.com/data_structures/postfix-evaluation.html>

Sample Input-1:

Postfix Expression 5 3 + 8 2 - \*

Sample Output-1:

Postfix Expression Evaluation Value is 48

Sample Input-2:

Postfix Expression 2 3 4 + \* 6 -

Sample Output-2:

Postfix Expression Evaluation Value is 8

**Program(Python):**

n=input("Enter any postfix operation:").split()

a=[]

for i in n:

if i.isdigit():

a.append(i)

else:

x=float(a.pop())

y=float(a.pop())

if ord(i)==43:

a.append(x+y)

elif ord(i)==45:

a.append(y-x)

elif ord(i)==42:

a.append(x\*y)

elif ord(i)==47:

a.append(y/x)

else:

print("Unknown Operation!")

exit()

print(f"Value of given postfix expression is {a[0]}")

**or**

n=input("Enter any postfix operation:").split()

a=[]

for i in n:

if i.isdigit():

a.append(i)

else:

x=(a.pop())

y=(a.pop())

a.append(str(eval(y+i+x)))

print(a[0])

**Output 1:**

Enter any postfix operation:2 3 4 + \* 6 -

Value of given postfix expression is 8.0

**Output 2:**

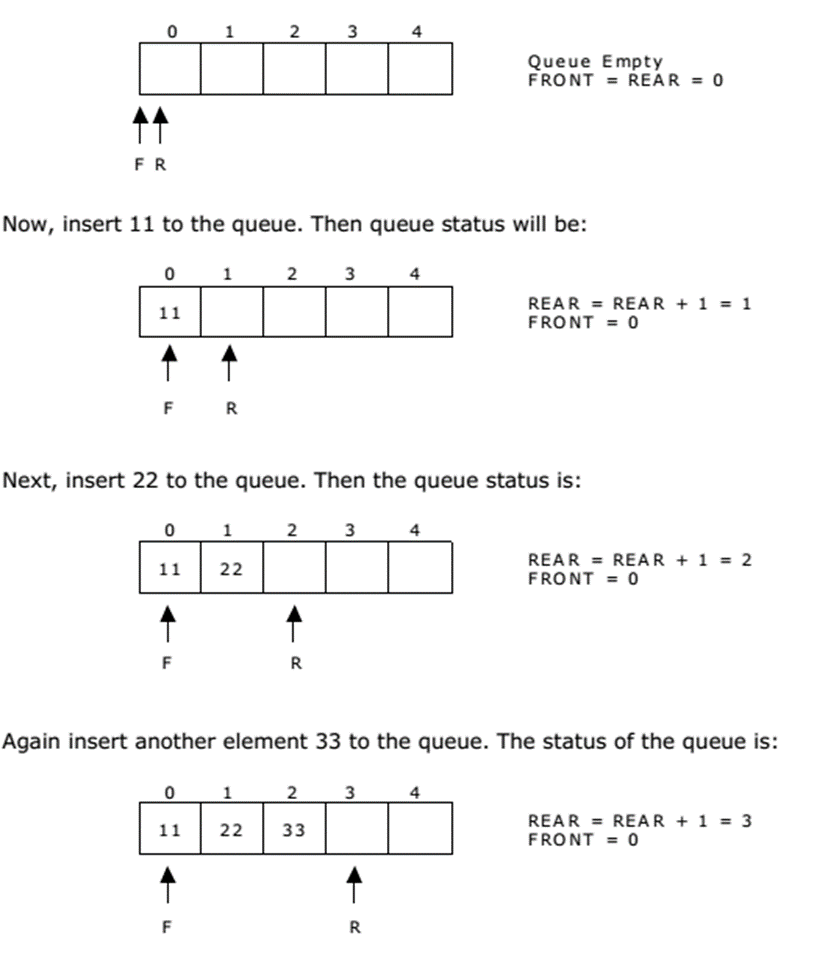
Enter any postfix operation:2 3 - 5 6 7 + / \*

Value of given postfix expression is -0.38461538461538464

## Task-2

We have discussed Queue operations called insert() and delete() in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to Queue (2) Delete element from Queue (3) Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases.

Use the following Example for Reference



**Program(C):**

#include <stdio.h>

#include<stdlib.h>

int rear=-1;

int front=-1;

int max\_size=6;

int a[20];

void insert(int x){

if(rear>max\_size)

printf("Queue Overflow\n");

else{

rear++;

a[rear]=x;

}

}

void delete(){

if(rear==front)

printf("Queue Empty\n");

else{

front++;

printf("Element deleted is %d\n",a[front]);

}

}

int display(){

if(rear==front)

printf("Queue Empty\n");

else{

printf("Elements of Queue are:");

for(int i=front+1;i<=rear;i++)

printf("%d ",a[i]);

printf("\n");

}

}

int main()

{

int m,a;

while(1){

printf("MENU\n1:insert 2:delete 3:Display 4:Exit\n");

scanf("%d",&m);

switch(m){

case 1:

printf("Enter a number to insert:\n");

scanf("%d",&a);

insert(a);

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

printf("Exiting Program...");

exit(0);

break;

}}

return 0;

}

**Output 1:**

MENU

1:insert 2:delete 3:Display 4:Exit

1

Enter a number to insert:

11

MENU

1:insert 2:delete 3:Display 4:Exit

1

Enter a number to insert:

12

MENU

1:insert 2:delete 3:Display 4:Exit

3

Elements of Queue are:11 12

MENU

1:insert 2:delete 3:Display 4:Exit

2

Element deleted is 11

MENU

1:insert 2:delete 3:Display 4:Exit

3

Elements of Queue are:12

MENU

1:insert 2:delete 3:Display 4:Exit

4

Exiting Program…

**Output 2:**

MENU

1:insert 2:delete 3:Display 4:Exit

2

Queue Empty

MENU

1:insert 2:delete 3:Display 4:Exit

3

Queue Empty

MENU

1:insert 2:delete 3:Display 4:Exit

1

Enter a number to insert:

12

MENU

1:insert 2:delete 3:Display 4:Exit

3

Elements of Queue are:12

MENU

1:insert 2:delete 3:Display 4:Exit

2

Element deleted is 12

MENU

1:insert 2:delete 3:Display 4:Exit

3

Queue Empty

MENU

1:insert 2:delete 3:Display 4:Exit

4

Exiting Program…

# WEEK 7

**Datastructures Lab Task II/IV B.Tech**

Topic: Applications of Stacks and Queues

Batch-1: 21-December-2021 @ 9.30 to 12

Batch-2: 23-December-2021 @ 8.40 to 11.10

## Task-1

In a class room we have discussed the stack operations called push() and pop(). Based on these operations design and implement the solution to identify the balanced and unbalanced expressions.

**Sample Input-1**

The expression like [(a+b)(a-b)]

**Sample output-1**

is known to be balanced

**Sample Input-2**

The expression like [(a+b)(a-b] is known to be unbalanced

**Sample output-2**

is known to be unbalanced

Design algorithm and rules for the same. Use a C/Python program for implementation.

**Program(Python):**

d={"(":")","[":"]","{":"}"}

a=[]

c=0

e=input().split()

for ele in e:

if ele in d.keys():

a.append(ele)

elif ele in d.values():

if d[a.pop()]==ele:

pass

else:

c=1

break

print("Balanced") if c==0 else print("Unbalanced")

**Output 1:**

Enter an Expression: [ [ ( { ] } )

Unbalanced

**Output 2:**

Enter an Expression: [ ( ) { } [ ] ]

Balanced

## TASK-2

We have discussed linked list operations called insert() and delete() in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to list (2) Delete element from list (3) Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases.

**Program(C):**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

}\*new,\*tos,\*p,\*temp;

void push(int val){

new=(struct node \*)malloc(sizeof(struct node));

new->data=val;

new->next=NULL;

if(tos==NULL)

tos=new;

else{

new->next=tos;

tos=new;

}

}

void pop(){

if(tos==NULL)

printf("Linked list Empty");

else{

temp=tos;

printf("Deleted element is %d",tos->data);

tos=tos->next;

free(temp);

}}

void display(){

p=tos;

if(tos == NULL)

printf("Linked list is empty\n");

else{

printf("Linked list Elements:");

while(p!=NULL){

printf(" %d",p->data);

p=p->next;

}

}}

int main(){

int m,a;

while(1){

printf("\nMENU\n1:Push 2:Pop 3:Display 4:Exit\n");

scanf("%d",&m);

switch(m){

case 1:

printf("Enter a number to push:\n");

scanf("%d",&a);

push(a);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

printf("Exiting Program...");

exit(0);

break;

}}

return 0;

}

**Output 1:**

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

12

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

123

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

123

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Deleted element is 123

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Linked list Elements: 123 12

MENU

1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

**Output 2:**

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Linked list Empty

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Linked list is empty

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

123

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

123

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Deleted element is 123

MENU

1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

# 

# 

# WEEK-8

**Datastructures Lab Task II/IV B.Tech**

Topic: Implementation of Stacks and Queues using linked lists

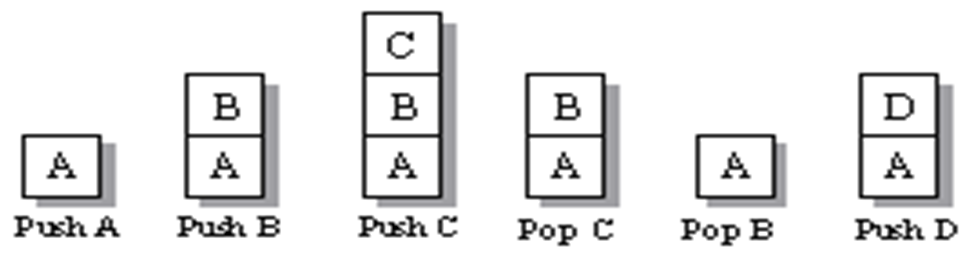
Batch-1: 21-December-2021 @ 9.30 to 12

Batch-2: 23-December-2021 @ 8.40 to 11.10

## Task-1

We have discussed stack operations called push () and pop () in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to stack (2) Delete element from stack (3) Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases. (Use Dynamic Implementation for character data)

Sample test case is as below:



**Program(C):**

// stack operations using Linked List

#include<stdio.h>

#include<stdlib.h>

struct node{

char data;

struct node \*next;

}\*new,\*tos,\*p,\*temp;

void push(char val){

new=(struct node \*)malloc(sizeof(struct node));

new->data=val;

new->next=NULL;

if(tos==NULL)

tos=new;

else{

new->next=tos;

tos=new;

}

}

void pop(){

if(tos==NULL)

printf("Stack Empty");

else{

temp=tos;

printf("Deleted element is %c",tos->data);

tos=tos->next;

free(temp);

}}

void display(){

p=tos;

if(tos == NULL)

printf("Stack is empty\n");

else{

printf("Stack Elements:");

while(p!=NULL){

printf(" %c",p->data);

p=p->next;

}

}}

int main(){

int m;

char a;

while(1){

printf("\nMENU\n1:Push 2:Pop 3:Display 4:Exit\n");

scanf("%d",&m);

switch(m){

case 1:

printf("Enter a number to push:\n");

scanf("%s",&a);

push(a);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

printf("Exiting Program...");

exit(0);

break;

}}

return 0;

}

**Output 1:**

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a letter to push:

a

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a letter to push:

b

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack Elements: b a

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Deleted element is b

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack Elements: a

MENU

1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

**Output 2:**

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Stack Empty

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack is empty

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a letter to push:

c

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Deleted element is c

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Stack Empty

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack is empty

MENU

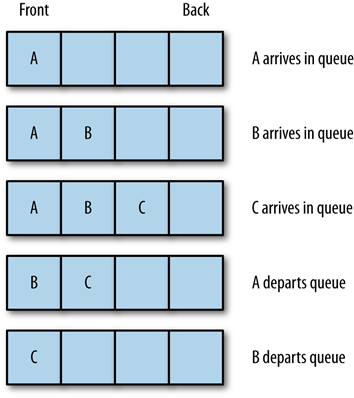
1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

## Task-2

We have discussed queue operations called insert () and delete () in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to Queue (2) Delete element from Queue (3)

Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases. (Use Dynamic Implementation for character dataImplementation for character data) 

**Program(C):**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node{

char data;

struct node \*next;

}\*new,

\*rear,

\*front,

\*dis,

\*temp;

void push(){

int i=1;

char value[100];

printf("enter the value to push:");

scanf("%s",&value[i]);

new=(struct node\*)malloc(sizeof(struct

node\*));

new->data=value[i];

i++;

new->next=NULL;

if (rear==NULL && front==NULL){

front=new;

rear=new;

}

else{

rear->next=new;

rear=new;

}

}

void pop(){

if (front==NULL )

{

printf("Queue is empty\n");

}

else{

temp=front;

printf("deleted element %c

\n",front->data);

front=front->next;

temp->next=NULL;

free(temp);

}

}

void display()

{

dis = front;

if(front == NULL)

{

printf("Queue is empty\n");

}

else

{

while(dis != NULL)

{

printf("%c ",dis -> data);

dis = dis -> next;

}

}

}

int main()

{

int option;

printf("1.push\n2.pop\n3.display\n4.stop\n");

while (1)

{

printf("enter the option:");

scanf("%d",&option);

switch(option){

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

exit(0);

}

}

return 0;

}

**Output 1:**

1.push

2.pop

3.display

4.stop

enter the option:1

enter the Letter to push:a

enter the option:1

enter the Letter to push:b

enter the option:1

enter the Letter to push:c

enter the option:2

deleted element a

enter the option:2

deleted element b

enter the option:3

c enter the option:4

**Output 2:**

1.push

2.pop

3.display

4.stop

enter the option:2

Queue is empty

enter the option:3

Queue is empty

enter the option:1

enter the Letter to push:a

enter the option:2

deleted element a

enter the option:2

Queue is empty

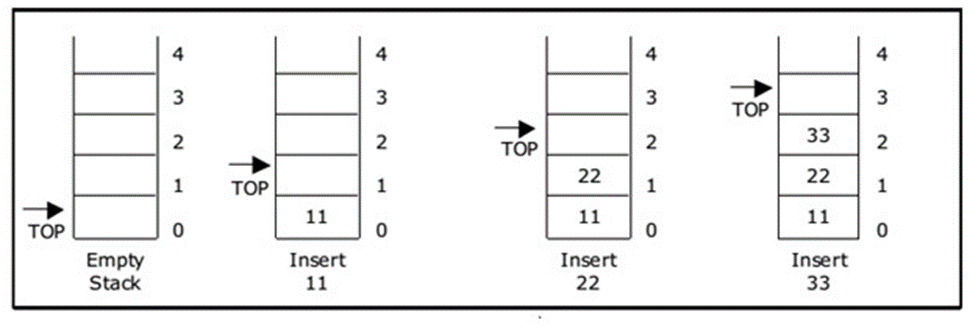
enter the option:3

Queue is empty

enter the option:4

## Task-3

We have discussed stack operations called push () and pop () in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to stack (2) Delete element from stack (3) Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases. (Use Dynamic Implementation for integer data)



**Program(C):**

// stack operations using Linked List

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

}\*new,\*tos,\*p,\*temp;

void push(int val){

new=(struct node \*)malloc(sizeof(struct node));

new->data=val;

new->next=NULL;

if(tos==NULL)

tos=new;

else{

new->next=tos;

tos=new;

}

}

void pop(){

if(tos==NULL)

printf("Stack Empty");

else{

temp=tos;

printf("Deleted element is %d",tos->data);

tos=tos->next;

free(temp);

}}

void display(){

p=tos;

if(tos == NULL)

printf("Stack is empty\n");

else{

printf("Stack Elements:");

while(p!=NULL){

printf(" %d",p->data);

p=p->next;

}

}}

int main(){

int m,a;

while(1){

printf("\nMENU\n1:Push 2:Pop 3:Display 4:Exit\n");

scanf("%d",&m);

switch(m){

case 1:

printf("Enter a number to push:\n");

scanf("%d",&a);

push(a);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

printf("Exiting Program...");

exit(0);

break;

}}

return 0;

}

**Output 1:**

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

11

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Deleted element is 11

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack is empty

MENU

1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

**Output 2:**

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Stack Empty

MENU

1:Push 2:Pop 3:Display 4:Exit

1

Enter a number to push:

12

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack Elements:12

MENU

1:Push 2:Pop 3:Display 4:Exit

2

Deleted element is 12

MENU

1:Push 2:Pop 3:Display 4:Exit

3

Stack is empty

MENU

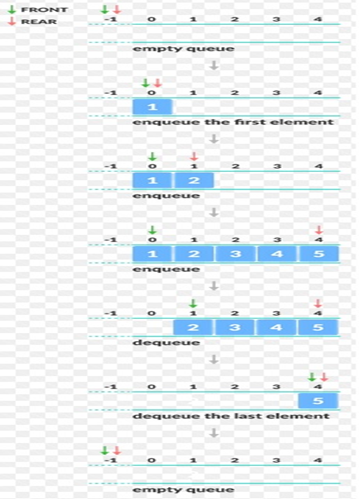
1:Push 2:Pop 3:Display 4:Exit

4

Exiting Program...

## Task-4

We have discussed queue operations called insert () and delete () in the class. Also illustrated with examples. Design and implement a menu driven C program with 4 operations. (1) Add element to Queue (2) Delete element from Queue (3) Traverse elements and (4) Exit. Write all possible examples supported by relevant test cases. (Use Dynamic Implementation for integer data)



**Program(C):**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

}\*new,\*rear,\*front,\*dis,\*temp;

void push(){

int value;

printf("enter the value to push:");

scanf("%d",&value);

new=(struct node\*)malloc(sizeof(struct node\*));

new->data=value;

new->next=NULL;

if (rear==NULL && front==NULL){

front=new;

rear=new;

}

else{

rear->next=new;

rear=new;

}

}

void pop(){

if (front==NULL )

{

printf("Queue is empty\n");

}

else{

temp=front;

printf("deleted element %d \n",front->data);

front=front->next;

temp->next=NULL;

free(temp);

}

}

void display()

{

dis = front;

if(front == NULL)

{

printf("Queue is empty\n");

}

else

{

while(dis != NULL)

{

printf("%d ",dis -> data);

dis = dis -> next;

}

}

}

int main()

{

int option;

printf("1.push\n2.pop\n3.display\n4.stop\n");

while (1)

{

printf("enter the option:");

scanf("%d",&option);

switch(option){

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

exit(0);

}

}

return 0;

}

**Output 1:**

1.push

2.pop

3.display

4.stop

enter the option:1

enter the value to push:12

enter the option:1

enter the value to push:13

enter the option:2

deleted element 12

enter the option:3

13 enter the option:4

**Output 2:**

1.push

2.pop

3.display

4.stop

enter the option:2

Queue is empty

enter the option:3

Queue is empty

enter the option:1

enter the value to push:12

enter the option:1

enter the value to push:13

enter the option:2

deleted element 12

enter the option:4

# WEEK-9

**Datastructures Lab Task II/IV B.Tech**

Topic: Applications of LinkedStacks&Queues

Batch-1: 11-January-2022 @ 9.30 to 12

Batch-2: 13- January -2022 @ 8.40 to 11.10

## Task-1

In the classroom we have discussed the algorithms to create the polynomial and addition of two polynomials . Based on the same discussion, implement a solution.

Input:

1st number = 5x2 + 4x1 + 2x0

2nd number = -5x1 - 5x0

Output:

5x2-1x1-3x0

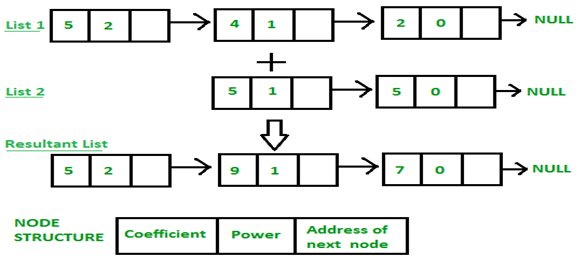
Input:

1st number = 5x3 + 4x2 + 2x0

2nd number = 5x^1 - 5x^0

Output:

5x3 + 4x2 + 5x1 - 3x0



**Program(C):**

#include <stdio.h>

#include<stdlib.h>

struct node{

int coef,expo;

struct node \*next;

}\*new1,\*p1,\*head1,\*new2,\*p2,\*head2;

int main(){

char option;

printf("Enter First Polynomial\n");

do

{

new1=(struct node\*)malloc(sizeof(struct node\*));

printf("Enter a term\n");

scanf("%d %d",&new1->coef,&new1->expo);

new1->next=NULL;

if(head1==NULL){

head1=new1;

p1=new1;

}

else{

p1->next=new1;

p1=new1;

}

printf("Do you want the next term Y/N");

scanf("%s",&option);

}while(option=='Y');

printf("Enter Second Polynomial\n");

do{

new2=(struct node\*)malloc(sizeof(struct node\*));

printf("Enter a term\n");

scanf("%d %d",&new2->coef,&new2->expo);

new2->next=NULL;

if(head2==NULL){

head2=new2;

p2=new2;

}

else{

p2->next=new2;

p2=new2;

}

printf("Do you want the next term Y/N");

scanf("%s",&option);

}while(option=='Y');

p1=head1;

p2=head2;

while(p1!=NULL && p2!=NULL){

if(p1->expo==p2->expo){

printf("%dx^%d",p1->coef+p2->coef,p1->expo);

p1=p1->next;

p2=p2->next;

}

else if(p1->expo > p2->expo){

printf("%dx^%d",p1->coef,p1->expo);

p1=p1->next;

}

else{

printf("%dx^%d",p2->coef,p2->expo);

p2=p2->next;

}

if(p1!=NULL || p2!=NULL)

printf("+");

}

return 0;

}

**Output:**

Enter First Polynomial

Enter a term

2 3

Do you want the next term Y/N Y

Enter a term

2 1

Do you want the next term Y/NY

Enter a term

2 0

Do you want the next term Y/NN

Enter Second Polynomial

Enter a term

2 4

Do you want the next term Y/NY

Enter a term

2 3

Do you want the next term Y/NY

Enter a term

2 2

Do you want the next term Y/NN

2x^4+4x^3+2x^2+

## Task-2

A polynomial p(x) is the expression in variable x which is in the form (axn + bxn-1 + …. + jx+ k), where a, b, c …., k fall in the category of real numbers and 'n' is non negative integer, which is called the degree of polynomial. Create a function that takes two sorted linked lists and merges them into a single sorted linked list. Your goal is to return a new linked list that contains the nodes from two lists in sorted order. You may assume the sort order is ascending. For example:

// list1

1 -> 4 -> 10 -> 11

// list2

-1 -> 2 -> 3 -> 6

// merged list

-1 -> 1 -> 2 -> 3 -> 4 -> 6 -> 10 -> 11

**Program(C):**

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* next;

};

void printList(struct Node\* head)

{

struct Node\* ptr = head;

while (ptr)

{

printf("%d —> ", ptr->data);

ptr = ptr->next;

}

printf("NULL\n");

}

void push(struct Node\*\* head, int data)

{

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

newNode->next = \*head;

\*head = newNode;

}

void moveNode(struct Node\*\* destRef, struct Node\*\* sourceRef)

{

if (\*sourceRef == NULL) {

return;

}

struct Node\* newNode = \*sourceRef;

\*sourceRef = (\*sourceRef)->next;

newNode->next = \*destRef;

\*destRef = newNode;

}

struct Node\* sortedMerge(struct Node\* a, struct Node\* b)

{

struct Node\* result = NULL;

struct Node\*\* lastPtrRef = &result;

while (1)

{

if (a == NULL)

{

\*lastPtrRef = b;

break;

}

else if (b == NULL)

{

\*lastPtrRef = a;

break;

}

if (a->data <= b->data) {

moveNode(lastPtrRef, &a);

}

else {

moveNode(lastPtrRef, &b);

}

lastPtrRef = &((\*lastPtrRef)->next);

}

return result;

}

int main(void)

{

int keys[] = { 1, 4, 10, 11};

int keys1[]={ -1, 2, 3, 6};

int n = sizeof(keys)/sizeof(keys[0]);

int n1= sizeof(keys1)/sizeof(keys1[0]);

struct Node \*a = NULL, \*b = NULL;

for (int i = n-1 ; i >= 0; i--) {

push(&a, keys[i]);

}

for (int j = n1-1; j >= 0; j--) {

push(&b, keys1[j]);

}

printf("First List: ");

printList(a);

printf("Second List: ");

printList(b);

struct Node\* head = sortedMerge(a, b);

printf("After Merge: ");

printList(head);

return 0;

}

## Task-3

Reverse a linked list: Given pointer to the head node of a linked list, the task is to reverse the linked list. We need to reverse the list by changing the links between nodes.

**Input**: Head of following linked list

1->2->3->4->NULL

**Output**: Linked list should be changed to,

4->3->2->1->NULL

**Program(C):**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

void create(struct node \*\*);

void reverse(struct node \*\*);

void release(struct node \*\*);

void display(struct node \*);

int main()

{

struct node \*p = NULL;

int n;

printf("Enter data into the list\n");

create(&p);

printf("Displaying the nodes in the list:\n");

display(p);

printf("Reversing the list...\n");

reverse(&p);

printf("Displaying the reversed list:\n");

display(p);

release(&p);

return 0;

}

void reverse(struct node \*\*head)

{

struct node \*p, \*q, \*r;

p = q = r = \*head;

p = p->next->next;

q = q->next;

r->next = NULL;

q->next = r;

while (p != NULL)

{

r = q;

q = p;

p = p->next;

q->next = r;

}

\*head = q;

}

void create(struct node \*\*head)

{

int c, ch;

struct node \*temp, \*rear;

do

{

printf("Enter number: ");

scanf("%d", &c);

temp = (struct node \*)malloc(sizeof(struct node));

temp->num = c;

temp->next = NULL;

if (\*head == NULL)

{

\*head = temp;

}

else

{

rear->next = temp;

}

rear = temp;

printf("Do you wish to continue [1/0]: ");

scanf("%d", &ch);

} while (ch != 0);

printf("\n");

}

void display(struct node \*p)

{

while (p != NULL)

{

printf("%d\t", p->num);

p = p->next;

}

printf("\n");

}

void release(struct node \*\*head)

{

struct node \*temp = \*head;

\*head = (\*head)->next;

while ((\*head) != NULL)

{

free(temp);

temp = \*head;

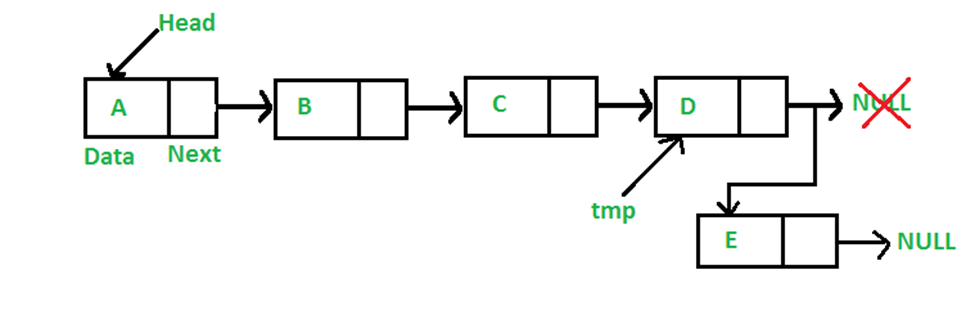
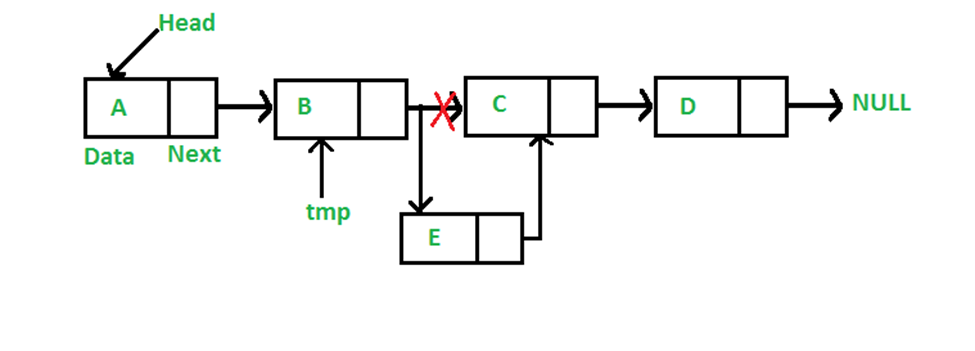
(\*head) = (\*head)->next;

}

}

## Task-4

Perform the following two operation for the case of single linked list

**Program(C):**

#include <stdio.h>

#include <stdlib.h>

struct node{

int val;

struct node \*next;

};

void print\_list(struct node \*head)

{

printf("H->");

while(head)

{

printf("%d->", head->val);

head = head->next;

}

printf("|||\n\n");

}

void insert\_front(struct node \*\*head, int value)

{

struct node \* new\_node = NULL;

new\_node = (struct node \*)malloc(sizeof(struct node));

if (new\_node == NULL)

{

printf("Failed to insert element. Out of memory");

}

new\_node->val = value;

new\_node->next = \*head;

\*head = new\_node;

}

void insert\_after(struct node \*head, int value, int after)

{

struct node \* new\_node = NULL;

struct node \*tmp = head;

while(tmp) {

if(tmp->val == after) { /\*found the node\*/

new\_node = (struct node \*)malloc(sizeof(struct node));

if (new\_node == NULL) {

printf("Failed to insert element. Out of memory");

}

new\_node->val = value;

new\_node->next = tmp->next;

tmp->next = new\_node;

return;

}

tmp = tmp->next;

}

}

void main()

{

int count = 0, i, val, after, before;

struct node \* head = NULL;

printf("Enter number of elements: ");

scanf("%d", &count);

for (i = 0; i < count; i++)

{

printf("Enter %dth element: ", i);

scanf("%d", &val);

insert\_front(&head, val);

}

printf("Initial List: ");

print\_list(head);

printf("Enter a value to insert in the list: ");

scanf("%d", &val);

printf("Insert after: ");

scanf("%d", &after);

insert\_after(head, val, after);

printf("List after insertion: ");

print\_list(head);

}

# WEEK 10:

**Datastructures Lab Task II/IV B.Tech**

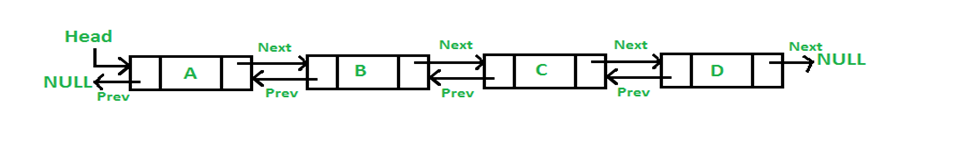
Topic: Types of Linked List (Double and Circular)

Batch-1: 25-January-2022 @ 9.30 to 12

Batch-2: 27- January -2022 @ 8.40 to 11.10

## Task-1

Design and Implement a program to create a doubly linked list. Use the following pseudocode. Justify your solution with all the possible test cases.



**Program(C):**

#include <stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*prev;

struct node \*next;

} \*new, \*head,\*p,\*temp;

void insert (int x)

{

new = (struct node \*) malloc (sizeof (struct node));

new->data = x;

new->next = NULL;

if (head == NULL){

head = new;

temp=new;}

else

{

new->prev = head;

head->next = new;

head=head->next;

}

}

void display(){

p=temp;

if(p == NULL)

printf("Linked List is empty\n");

else{

printf("Linked List Elements:");

while(p!=NULL){

printf(" %d",p->data);

p=p->next;

}

printf("\n");

}

}

int main(){

int element;

char choice;

do{

printf("Enter element to insert");

scanf("%d",&element);

insert(element);

display();

printf("do you need one more term(Y/N)");

scanf("%s",&choice);

}

while(choice=='Y');

return 0;

}

**Output 1:**

Enter element to insert1

Linked List Elements: 1

do you need one more term(Y/N)Y

Enter element to insert12

Linked List Elements: 1 12

do you need one more term(Y/N)N

**Output 2:**

Enter element to insert11

Linked List Elements: 11

do you need one more term(Y/N)Y

Enter element to insert12

Linked List Elements: 11 12

do you need one more term(Y/N)Y

Enter element to insert13

Linked List Elements: 11 12 13

do you need one more term(Y/N)Y

Enter element to insert14

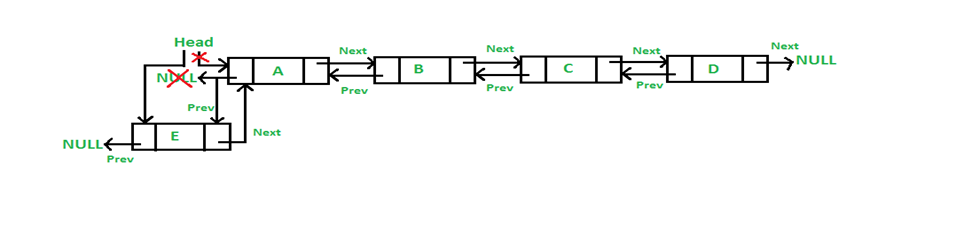
Linked List Elements: 11 12 13 14

do you need one more term(Y/N)N

## 

## Task-2

Design and Implement a program to insert a node in the beginning of the double linked list. Use the following pseudocode. Justify your solution with all the possible test cases. (Use the code designed in task-1 and make extension).



**Program(C):**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*prev;

struct node \*next;

} \* new, \*head, \*p, \*temp;

void insert(int x)

{

new = (struct node \*)malloc(sizeof(struct node));

new->data = x;

new->prev = NULL;

if (head == NULL)

{

head = new;

temp = new;

}

else

{

new->next = head;

head->prev = new;

head = head->prev;

}

}

void display()

{

p = temp;

if (p == NULL)

printf("Linked List is empty\n");

else

{

printf("Linked List Elements:");

while (p != NULL)

{

printf(" %d", p->data);

p = p->prev;

}

printf("\n");

}

}

int main()

{

int element;

char choice;

do

{

printf("Enter element to insert");

scanf("%d", &element);

insert(element);

display();

printf("do you need one more term(Y/N)");

scanf("%s", &choice);

} while (choice == 'Y');

return 0;

}

**Output :**

Enter element to insert1

Linked List Elements: 1

do you need one more term(Y/N)Y

Enter element to insert2

Linked List Elements: 1 2

do you need one more term(Y/N)Y

Enter element to insert4

Linked List Elements: 1 2 4

do you need one more term(Y/N)Y

Enter element to insert6

Linked List Elements: 1 2 4 6

do you need one more term(Y/N)Y

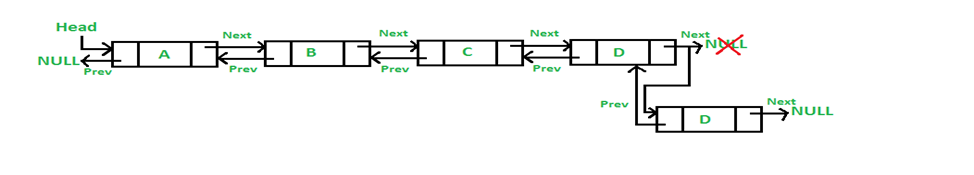
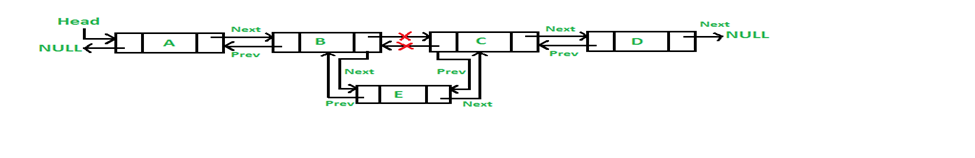
Enter element to insert8

Linked List Elements: 1 2 4 6 8

do you need one more term(Y/N)N

## Task-3

Based on the knowledge acquired from Task-1 and Task-2. Perform (i) Insert of node at the end of a double linked list and (ii) Insert of node at the specified position of a double linked list.



**Program(C):**

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*prev;

struct node \*next;

int data;

};

struct node \*head;

void insertion\_last();

void insertion\_specified();

void display();

void main ()

{

int choice =0;

printf("\n1.Insert at last\n2.Insert at any random location\n3.Show\n4.Exit\n");

while(choice != 9)

{

printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

insertion\_last();

break;

case 2:

insertion\_specified();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

void insertion\_last()

{

struct node \*ptr,\*temp;

int item;

ptr = (struct node \*) malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value");

scanf("%d",&item);

ptr->data=item;

if(head == NULL)

{

ptr->next = NULL;

ptr->prev = NULL;

head = ptr;

}

else

{

temp = head;

while(temp->next!=NULL)

{

temp = temp->next;

}

temp->next = ptr;

ptr ->prev=temp;

ptr->next = NULL;

}

}

}

void insertion\_specified()

{

struct node \*ptr,\*temp;

int item,loc,i;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\n OVERFLOW");

}

else

{

temp=head;

printf("Enter the location");

scanf("%d",&loc);

for(i=0;i<loc;i++)

{

temp = temp->next;

if(temp == NULL)

{

printf("\n There are less than %d elements", loc);

return;

}

}

printf("Enter value");

scanf("%d",&item);

ptr->data = item;

ptr->next = temp->next;

ptr -> prev = temp;

temp->next = ptr;

temp->next->prev=ptr;

}

}

void display()

{

struct node \*ptr;

printf("\n printing values...\n");

ptr = head;

while(ptr != NULL)

{

printf("%d\n",ptr->data);

ptr=ptr->next;

}

}

## Task-4

Design and Implement a program to (i) create circular linked list node (ii) create circular linked list (iii) Insert elements at different positions. Use the knowledge acquired in Task-1,2 and 3. Justify your solution with all the possible test cases.

**Program(C):**

#include<stdio.h>

#include<stdlib.h>

struct node

{

struct node \*prev;

struct node \*next;

int data;

};

struct node \*head;

void insertion\_beginning();

void insertion\_last();

void display();

void main ()

{

int choice =0;

printf("\n1.Insert in Beginning\n2.Insert at last\n3.Show\n4.Exit\n");

while(choice != 9)

{

printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

insertion\_beginning();

break;

case 2:

insertion\_last();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

void insertion\_beginning()

{

struct node \*ptr,\*temp;

int item;

ptr = (struct node \*)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter Item value");

scanf("%d",&item);

ptr->data=item;

if(head==NULL)

{

head = ptr;

ptr -> next = head;

ptr -> prev = head;

}

else

{

temp = head;

while(temp -> next != head)

{

temp = temp -> next;

}

temp -> next = ptr;

ptr -> prev = temp;

head -> prev = ptr;

ptr -> next = head;

head = ptr;

}

}

}

void insertion\_last()

{

struct node \*ptr,\*temp;

int item;

ptr = (struct node \*) malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value");

scanf("%d",&item);

ptr->data=item;

if(head == NULL)

{

head = ptr;

ptr -> next = head;

ptr -> prev = head;

}

else

{

temp = head;

while(temp->next !=head)

{

temp = temp->next;

}

temp->next = ptr;

ptr ->prev=temp;

head -> prev = ptr;

ptr -> next = head;

}

}

}

void display()

{

struct node \*ptr;

ptr=head;

if(head == NULL)

{

printf("\nnothing to print");

}

else

{

printf("\n printing values ... \n");

while(ptr -> next != head)

{

printf("%d\n", ptr -> data);

ptr = ptr -> next;

}

printf("%d\n", ptr -> data);

}

}

# WEEK 11:

**Datastructures Lab Task II/IV B.Tech**

Topic: Applications of LinkedStacks&Queues

Batch-1: 01-February-2022 @ 9.30 to 12

Batch-2: 03- February -2022 @ 8.40 to 11.10

## Task-1

In the classroom we have discussed the algorithms for different traversal techniques like (i) Inorder (ii) preorder and (iii) postorder. Based on the same discussion, implement a solution. Use the following LOC and complete the implementation part. Also run all the possible test cases.

**Program(C):**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int num;

struct node \*left;

struct node \*right;

};

struct node \*create ()

{

struct node \*temp;

temp = (struct node \*) malloc (sizeof (struct node));

temp->left = NULL;

temp->right = NULL;

printf ("Enter value:");

scanf ("%d", &temp->num);

return temp;

}

void inorder (struct node \*T)

{

if (T != NULL)

{

inorder (T->left);

printf("%d",T->num);

inorder (T->right);

}

}

void preorder (struct node \*T)

{

if (T != NULL)

{

printf("%d", T->num);

preorder (T->left);

preorder (T->right);

}

}

void postorder (struct node \*T)

{

if (T != NULL)

{

postorder (T->left);

postorder (T->right);

printf("%d", T->num);

}

}

void main()

{

struct node \*v1, \*v2, \*v3, \*v4, \*v5, \*v6, \*v7;

v1 = create();

v2 = create();

v3 = create();

v4 = create();

v5 = create();

v6 = create();

v7 = create();

v1->left = v2;

v1->right = v3;

v2->left = v4;

v2->right = v5;

v3->left = v6;

v3->right = v7;

printf ("\nInorder:");

inorder (v1);

printf ("\nPreorder:");

preorder (v1);

printf ("\nPostorder:");

postorder (v1);

}

**Output 1:**

Enter value:1

Enter value:2

Enter value:3

Enter value:4

Enter value:5

Enter value:6

Enter value:7

Inorder:4251637

Preorder:1245367

Postorder:4526731

**Output 2:**

Enter value:2

Enter value:3

Enter value:1

Enter value:4

Enter value:5

Enter value:6

Enter value:7

Inorder:4352617

Preorder:2345167

Postorder:4536712

## Task-2

In the classroom we have discussed the algorithms for infix to postfix conversion. Based on the same discussion, implement a solution. Also run all the possible test cases.

**Testcase-1**

**Infix Expression : 3+4\*5/6**

**Postfix Expression : 3 4 5 \* 6 / +**

**Testcase-2**

**Infix Expression :** A+(B\*C-(D/E^F)\*G)\*H

**Postfix Expression :** ABC\*DEF^/G\*-H\*+

**Program(C):**

#include<stdio.h>

#include<string.h>

#include<math.h>

#include<stdlib.h>

#define MAX 50

void push(long int symbol);

long int pop();

void infix\_to\_postfix();

int priority(char symbol);

int isEmpty();

int white\_space(char);

char infix[MAX], postfix[MAX];

long int stack[MAX];

int top;

int main()

{

top=-1;

printf("Enter infix : ");

fgets(infix,50,stdin);

infix\_to\_postfix();

printf("Postfix : %s\n",postfix);

return 1;

}

void infix\_to\_postfix()

{

unsigned int i,p=0;

char next;

char symbol;

for(i=0;i<strlen(infix);i++)

{

symbol=infix[i];

if(!white\_space(symbol))

{

switch(symbol)

{

case '(':

push(symbol);

break;

case ')':

while((next=pop())!='(')

postfix[p++] = next;

break;

case '+':

case '-':

case '\*':

case '/':

case '%':

case '^':

while( !isEmpty( ) && priority(stack[top])>= priority(symbol) )

postfix[p++]=pop();

push(symbol);

break;

default:

postfix[p++]=symbol;

}

}

}

while(!isEmpty( ))

postfix[p++]=pop();

postfix[p]='\0';

}

int priority(char symbol)

{

switch(symbol)

{

case '(':

return 0;

case '+':

case '-':

return 1;

case '\*':

case '/':

case '%':

return 2;

case '^':

return 3;

default :

return 0;

}

}

void push(long int symbol)

{

if(top>MAX)

{

printf("Stack overflow\n");

exit(1);

}

stack[++top]=symbol;

}

long int pop()

{

if( isEmpty() )

{

printf("Stack underflow\n");

exit(1);

}

return (stack[top--]);

}

int isEmpty()

{

if(top==-1)

return 1;

else

return 0;

}

int white\_space(char symbol)

{

if( symbol == ' ' || symbol == '\t' )

return 1;

else

return 0;

}

**Output 1:**

Enter infix : A+(B\*C-(D/E^F)\*G)\*H

Postfix : ABC\*DEF^/G\*-H\*+

**Output 2:**

Enter infix : 3+4\*5/6

Postfix : 345\*6/+

# WEEK 12:

**Datastructures Lab Task II/IV B.Tech**

Topic: Implementation of Different types of Trees

Batch-1: 08-February-2022 @ 9.30 to 12

Batch-2: 10- February -2022 @ 8.40 to 11.10

## Task-1

Design, develop and execute a program in C/Python to implement a Binary Search tree where each node consists of integers. The program should support the following functions and show different test cases.

a. Create a Binary Search tree

b. Insert a new node

c. Delete a node if it is found, otherwise display appropriate message

d. Display the nodes of Binary Search tree

e. Validate Binary Search tree

**Program(C):**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct node

{

int data;

struct node \*left;

struct node \*right;

};

struct node \*create(int data)

{

struct node \*new;

new = (struct node \*)malloc(sizeof(struct node));

new->data = data;

new->left = NULL;

new->right = NULL;

return new;

}

// Inorder Traversal

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

printf("%d ", root->data);

inorder(root->right);

}

}

// BST Validation

int isBSTcheck(struct node \*root, int min, int max)

{

if (root == NULL)

return 1;

if (root->data <= min || root->data >= max)

return 0;

return isBSTcheck(root->left, min, root->data) && isBSTcheck(root->right, root->data, max);

}

int isBST(struct node \*root)

{

return isBSTcheck(root, INT\_MIN, INT\_MAX);

}

// BST Insertion

struct node \*insert(struct node \*root, int value)

{

if (root == NULL)

return create(value);

if (value < root->data)

root->left = insert(root->left, value);

else if (value > root->data)

root->right = insert(root->left, value);

}

// BST Deletion

struct node \*inOrderPredecessor(struct node \*root)

{

root = root->left;

while (root->right != NULL)

{

root = root->right;

}

return root;

}

struct node \*deleteNode(struct node \*root, int value)

{

struct node \*iPre;

if (root == NULL)

{

return NULL;

}

if (root->left == NULL && root->right == NULL)

{

free(root);

return NULL;

}

if (value < root->data)

{

root->left = deleteNode(root->left, value);

}

else if (value > root->data)

{

root->right = deleteNode(root->right, value);

}

else

{

iPre = inOrderPredecessor(root);

root->data = iPre->data;

root->left = deleteNode(root->left, iPre->data);

}

return root;

}

int main()

{

/\* 5

/ \

3 6

/ \

2 4

\*/

struct node \*p = create(5);

struct node \*p1 = create(3);

struct node \*p2 = create(6);

struct node \*p3 = create(2);

struct node \*p4 = create(4);

p->left = p1;

p->right = p2;

p1->left = p3;

p1->right = p4;

int option, element;

printf("1.Insert\n2.Delete\n3.display\n4.Validate\n5.Stop\n");

while (1)

{

printf("enter the option:");

scanf("%d", &option);

switch (option)

{

case 1:

{

printf("Enter Element to Insert:");

scanf("%d", &element);

insert(p, element);

break;

}

case 2:

{

printf("Enter Element to Delete:");

scanf("%d", &element);

deleteNode(p, element);

break;

}

case 3:

{

inorder(p);

printf("\n");

break;

}

case 4:

{

if (isBST(p))

printf("Yes!It is a BST\n");

else

printf("Yes!It is not a BST\n");

break;

}

case 5:

printf("Exiting Program.....");

exit(0);

}

}

return 0;

}

**OUtput 1:**

1.Insert

2.Delete

3.display

4.Validate

5.Stop

enter the option:1

Enter Element to Insert:1

enter the option:3

1 2 3 4 5 6

enter the option:2

Enter Element to Delete:1

enter the option:3

2 3 4 5 6

enter the option:2

Enter Element to Delete:2

enter the option:3

3 4 5 6

enter the option:4

Yes!It is a BST

enter the option:5

Exiting Program.....

## Task-2

AVL tree is a descendant of Binary Search Tree but overcomes its drawback of increasing complexity if the elements are sorted. It monitors the balance factor of the tree to be 0 or 1 or -1. In case the tree becomes unbalanced, corresponding rotation techniques are performed to balance the tree. Implement insert operation for the creation of nodes in AVL tree. Check the balance factor. If not satisfied, apply rotations.

**1. Left Rotation:** If the addition of a node to the tree’s right makes it imbalance then, in that case, Left Rotation needs to be performed.

**2. Right Rotation:** If the addition of a node to the left of the tree makes the node imbalance then Right Rotation needs to be performed. In other words, When the number of nodes increases on the left side, then there emerges a need to shift the elements to the right side to balance it thus it is said to be Right Rotation.

**3. Left-Right Rotation:** This type of rotation is a combination of the above 2 rotations explained. This type of rotation occurs when one element is added to the right subtree of a left tree.

In such a case first, perform left rotation on the subtree followed by a right rotation of the left tree.

**4. Right-Left Rotation:** This type of rotation is also composed of a sequence of above 2 rotations. This type of rotation is needed when an element is added to the left of the right subtree, and the tree becomes imbalanced. In such a case, we first perform right rotation on the right subtree and then left rotation on the right tree.

Justify the program with different test cases.

**Program(C):**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int key;

struct Node \*left;

struct Node \*right;

int height;

};

int max(int a, int b);

int height(struct Node \*N)

{

if (N == NULL)

return 0;

return N->height;

}

int max(int a, int b)

{

return (a > b)? a : b;

}

struct Node\* newNode(int key)

{

struct Node\* node = (struct Node\*)malloc(sizeof(struct Node));

node->key = key;

node->left = NULL;

node->right = NULL;

node->height = 1;

return(node);

}

struct Node \*rightRotate(struct Node \*y)

{

struct Node \*x = y->left;

struct Node \*T2 = x->right;

x->right = y;

y->left = T2;

y->height = max(height(y->left), height(y->right))+1;

x->height = max(height(x->left), height(x->right))+1;

return x;

}

struct Node \*leftRotate(struct Node \*x)

{

struct Node \*y = x->right;

struct Node \*T2 = y->left;

y->left = x;

x->right = T2;

x->height = max(height(x->left), height(x->right))+1;

y->height = max(height(y->left), height(y->right))+1;

return y;

}

int getBalance(struct Node \*N)

{

if (N == NULL)

return 0;

return height(N->left) - height(N->right);

}

struct Node\* insert(struct Node\* node, int key)

{

/\* 1. Perform the normal BST insertion \*/

if (node == NULL)

return(newNode(key));

if (key < node->key)

node->left = insert(node->left, key);

else if (key > node->key)

node->right = insert(node->right, key);

else // Equal keys are not allowed in BST

return node;

/\* 2. Update height of this ancestor node \*/

node->height = 1 + max(height(node->left),height(node->right));

int balance = getBalance(node);

// Left Left Case

if (balance > 1 && key < node->left->key)

return rightRotate(node);

// Right Right Case

if (balance < -1 && key > node->right->key)

return leftRotate(node);

// Left Right Case

if (balance > 1 && key > node->left->key)

{

node->left = leftRotate(node->left);

return rightRotate(node);

}

// Right Left Case

if (balance < -1 && key < node->right->key)

{

node->right = rightRotate(node->right);

return leftRotate(node);

}

return node;

}

void preOrder(struct Node \*root)

{

if(root != NULL)

{

printf("%d ", root->key);

preOrder(root->left);

preOrder(root->right);

}

}

int main()

{

struct Node \*root = NULL;

int ch,x;

do

{

printf("enter your node");

scanf("%d",&x);

root = insert(root, x);

printf("would like to enter another node 1/0:");

scanf("%d",&ch);

}while(ch==1);

printf("Preorder traversal of the constructed AVL tree is \n");

preOrder(root);

return 0;

}