LABORATORY MANUAL

ON

DATA STRUCTURES AND ALGORITHMS LAB

II YEAR I SEM

Prepared by

Dr. K. JAYARAJAN

Professor

DEPARTMENT OF IT & CSIT

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

Autonomous Institution, UGC, Govt. of India
Programmes Accredited by NBA
Accredited by NAAC with A+ Grade
Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institute
Maisammaguda (V), Dhullapally (Post), (Via) Kompally,
Medchal Malkajgiri Dist. T.S-500100
2024 – 2025

VISION AND MISSION OF THE INSTITUTION

VISION

To become self-sustainable institution which is recognized for its new age engineering through innovative teaching and learning culture, inculcating research and entrepreneurial ecosystem, and sustainable social impact in the community.

MISSION

- To offer undergraduate and post-graduate programs that is supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
- To provide necessary support structures for students, which will contribute to their personal and professional growth and enable them to become leaders intheir respective fields.
- To provide faculty and students with an ecosystem that fosters research and development through strategic partnerships with government organizations and collaboration with industries.
- To contribute to the development of the region by using our technological expertise to work with near by communities and support them in their social and economic growth.

VISION AND MISSION OF IT DEPARTMENT

IT VISION

❖ To emerge as a center of excellence in the department of IT is to empower students with new wave technologies to produce technically proficient and accomplished intellectual IT professionals specifically to meet the modern challenges of the contemporary computing industry and society

❖ Providing the students with most conducive academic environment and making them towards serving the society with advanced technologies

IT MISSION

❖ M1: The mission of the department of Information Technology is to afford excellence education for students, in the conventional and modern areas of information technology and build up students with high-quality principled trainings, thus manifesting their global personality development.

❖ M2: To impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise

❖ M3: Training the students into competent and confident world class professionals with excellent technical and communication skills.

❖ M4: To provide quality education through innovative teaching and learning process that yields advancements in state-of-the-art information technology.

❖ M5:To inculcate the spirit of ethical values contributing to the welfare of the society by offering courses in the curriculum design.

Program Educational Objectives (PEOs):

PEO 1 INTUITION: To make students competent for higher studies and employable, to meet industrial requirements.

PEO 2 CORE COMPETENCE: To develop students having core competence in science, mathematics, statistics and fundamentals of Data Science to address ever changing industrial requirements globally.

PEO 3 BREADTH: To create academically conducive environment to learn engineering skills in the domains such as Data Analytics, Data Modelling, Data Visualization and Allied Technologies.

PEO 4 PROFESSIONALISM: To enrich students with professional ethics, leadership qualities, and entrepreneurial skills.

PEO 5 LIFE LONG LEARNING: An ability to engage in lifelong learning for effective adaptation to technological developments.

Program Outcomes (POs):

Program Outcomes	
PO1	**Engineering knowledge**: Applythe knowledge of Mathematics, Science, Engineering Fundamentals, and an Engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, Formulate, Review Research Literature, and Analyze Complex Engineering Problems Reaching substantiated conclusions using first principles of Mathematics, Natural Sciences, and Engineering Sciences.
PO3	**Design/development of solutions**: Design Solutions for Complex Engineering problems and design system components or processes that meet the specified needswith appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use Research-Based Knowledge and Research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage**: Create, Select, and Apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the eliminations.
PO6	**The engineer and society**: Apply reasoning informed bythe contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate theknowledge of, and need for sustainable development.
PO8	**Ethics**: Applyethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehendand write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of theengineering and management principles and apply these to one’s own work,as amember and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes(PSOs):

PSO 1: An ability to analyze a problem, design algorithm, identify and define the computing requirements within realistic constraints in multidisciplinary areas by understanding the core principles and concepts of Information Technology.

PSO 2: Knowledge of data management system like data acquisition, big data so as to enable students in solving problems using the techniques of data analytics like pattern recognition and knowledge discovery.

PSO 3: Effectively integrate IT based solutions into the user environment.

COURSE NAME: DATA STRUCTURES AND ALGORITHMS LAB

COURSE CODE: 2205PC61

COURSE OBJECTIVES:

Upon successful completion of this Lab the student will be able to:

- To make the student to implement data structures using python and C programming languages.
- To make the student write ADTS for all data structures.

COURSE OUTCOMES:

1. For a given algorithm student will able to analyze the algorithms to determine time & computation complexity and justify the correctness.
2. For a given Search problem (Linear Search and Binary Search) student will able to implement it.
3. For a given problem of Stacks, Queues and linked list student will able to implement it and analyze the same to determine the time and computation complexity.
4. Student will able to write an algorithm Selection Sort, Bubble Sort, Insertion Sort, Quick Sort, Merge Sort, Heap Sort and compare their performance in term of Space and Time complexity.

COURSE OUTCOME'S MAPPING WITH PROGRAM OUTCOME'S:

Program Outcomes	
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and abilityto engage in independent and life-long learning in the broadest context of technological change.

COURSE OUTCOME'S MAPPING WITH PROGRAM OUTCOME'S:

1805PC31	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO 1	✔	✔	✔		✔						✔	
CO 2	✔		✔	✔		✔				✔		
CO 3			✔	✔		✔			✔		✔	
CO 4		✔	✔		✔	✔					✔	

INDEX

WEEK-1

Aim: Write a C program and Python program to implement the following searching techniques in both recursive and non recursive manner.

i)Linear Search ii) Binary Search.

//Program to implement linear search using recursion

```
#include<stdio.h>
int lsearch_recursion(int a[],int n,int key);
int main()
{
int a[10],i,key,n,l;
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter elements into an array\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter key to search\n");
scanf("%d",&key);
l=lsearch_recursion(a,n,key);
printf("Element found at location %d",l+1);return 0;
}
int lsearch_recursion(int a[],int n,int key)
{ if(n<0)
return -1;
if(key==a[n-1])
return n-1;
 lsearch_recursion(a,n-1,key);  }
```

Output:

```
Enter number of elements 4

 Enter elements into an array 23 45
21

12

Enter    Key    to    Search    45
Element found at location 2
```

//Program to implement Linear search using non-recursion

```
#include<stdio.h>
void lsearch(int a[],int n,int key);
int main()
{
int a[10],i,key,n,l;
printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enterelements into an array\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enterkey to search\n");
scanf("%d",&key);
lsearch(a,n,key);
return 0;}
void lsearch(int a[],int n,int key)
{
int i,flag=0;
for(i=0;i<n;i++)
{
```

```
if(a[i]==key)
{
flag=1;
 break;
}}
if(flag==1)
printf("Elementfound at location %d\n",i+1);
else
printf("Element not found\n");
}
```

Output:

```
Enter  number  of  elements  5
Enter elements into an array 12
56
78
45
34
Enter    key    to    search    34
Element found at location 5
```

//Program to implement linear search iteratively using Python

```
def search(arr, x):
for i in range(len(arr)):
if arr[i] == x:
return i

return -1
search_arr = [5, 15, 6, 9, 4]
key = 4
ans = search(search_arr, key)
if(ans != -1):
print("The element", key, "is found at", ans, "index of the given array")
else:
print('Key is not present')
```

output:

The ele 4 is found at 4 index of the givenarray

//Program to implement linear search recursively using Python

```
def linear_search(arr, size, key):
# If the array is empty we will return -1
if (size == 0):
return -1

elif (arr[size - 1] == key):
# Return the index of found key.
return size - 1

return linear_search(arr, size - 1, key)
arr = [5, 15, 6, 9, 4]
key = 4
size = len(arr)

# Calling the Function
ans = linear_search(arr, size, key)
if ans != -1:
print("The element ",key," is found  at",ans,"index of the given array.")
```

```
else:
print("The element", key,"is not found")
```

//Program to implement Binary search using recursion

```
#include<stdio.h>
int binsearch(int a[], int x, int low, int high)
{
int mid;
if (low > high)
return -1;
mid = (low + high) / 2;
 if (x == a[mid])
{ return (mid);
}
else if (x < a[mid]){
binsearch(a, x, low, mid - 1);
}
else
{
binsearch(a, x, mid + 1, high);
}
}
int main()
{
int a[10],i,n,key,pos;
printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enterelements into an array\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter key element to be search\n");
 scanf("%d",&key);
 pos=binsearch(a,key,0,n-1);
printf("Key is found at position %d \n",pos+1);

 return 0;}
```

Output:

```
Enter  number  of  elements  6
Enter elements into an array 23
12
2

34

56

90

Enter key element to be search 2
Key is found at position 3
```

//Program to implement Binary search using non-recursion

```
#include<stdio.h>

int bsearch(int a[],int n,int key);

 int main(){

int a[10],i,n,key,pos;

printf("Enter number of elements\n");

scanf("%d",&n);
printf("Enter elements into an array\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter key element to be search\n");
scanf("%d",&key);
pos=bsearch(a,n,key);
printf("Key is found at position %d \n",pos+1);
return 0;}

int bsearch(int a[],int n,int key)

{

int low=0;

int high=n-1;

 int mid;
while(low<=high){
mid=(low+high)/2;
if(key==a[mid])
return mid;
else
if(key<mid)
high=mid-1;
else
low=mid+1;
}//while

}//function
```

Output:

```
Enter number of elements 5

Enter elements into an array12
24
35

36

89

Enter keyelement to be search 36
Key is found at position 4
```

# iterative Binary Search in python

```
def binarySearch(array, x, low, high):
    # Repeat until the pointers low and high meet each other
    while low <= high:
        mid = low + (high - low)//2
        if array[mid] == x:
            return mid
        elif array[mid] < x:
            low = mid + 1
        else:
            high = mid - 1
    return -1
array = [3, 4, 5, 6, 7, 8, 9]
x = 4
result = binarySearch(array, x, 0, len(array)-1)
if result != -1:
    print("Element is present at index " + str(result))
else:
print("Not found")
```

program for recursive binary search using python

```
def binary_search(arr, low, high, x):
        # Check base case
        if high >= low:
                mid = (high + low) // 2
                # If element is present at the middle itself
                if arr[mid] == x:
                        return mid
                # If element is smaller than mid, then it can only
                # be present in left subarray
                elif arr[mid] > x:
```

```
            return binary_search(arr, low, mid - 1, x)

        # Else the element can only be present in right subarray
        else:
            return binary_search(arr, mid + 1, high, x)
    else:   # Element is not present in the array
        return -1
```

WEEK-2

Aim: Write C programs to implement the following using an array.

a) Stack ADT b) Queue ADT

//Program to implement Stack ADT

```
#include<stdio.h>

#define SIZE 50

void push(int);

void pop();

void display();

void peek();

int stack[20],top=-1;

void main()

{

int ch,ele;

while(1){

printf("1.push\n2.pop\n3.display\n4.peek\n5.Exit");

printf("enter choice");

scanf("%d",&ch);

switch(ch){

case 1:printf("enter ele");

        scanf("%d",&ele);
        push(ele);

        break;

case 2:pop();

        break;

case 3:display();

        break;

case 4:peek();

        break;

case 5:exit(0);

default:
printf("invalid  choice\n");

}}}

void push(int ele)

{

if(top>=SIZE-1)

printf("stack overflow\n");

else{

top++;

stack[top]=ele;

}}

void pop(){

if(top==-1)

printf("stack underflow\n");
```

```
else{

printf("the deleted element is %d\n",stack[top]);

top--;

printf("deleted successfully\n");

}}

void display()

{

int i;

if(top==-1)

printf("stack is empty(underflow)\n");

else{

for(i=top;i>=0;i--)

printf("->%d",stack[i]);

}}

void peek()

{

if (top==-1)

printf("stack is empty\n")

else

printf("the top element of the stack is %d\n",stack[top]);
        }
```

Output:

```
Stack 1.Push 2.Pop 3.Display 4.peek5.Exit

 Enter your Choice:1

Enter Element:34

Elements(34) has been pushed at 0 Stack

Enter your Choice:1 Enter Element:67

Element(67) has been pushed at 1 Stack

Enter your Choice:3 34 67

 Enter your Choice:2

Element has been popped out! Stack

Enter your Choice:3 34

Enter your Choice:5
```

#stack using python program

```
stack=[]
def push():
    if len(stack)==n:
        print("list is full!!")
    else:
        element=input("enter the element")
        stack.append(element)
        print(stack)
def pop_element():
    if not stack:
        print("stack is empty")
    else:
        e=stack.pop()
        print("removed element :",e)
        print(stack)
def display():
    if not stack:
        print("stack is empty")
    else:
     print(stack)
    print("Top most element is : ",stack[0])
n=int(input("limit of stack : "))
while (1):
    print("select the operation \n1.push\n2.pop\n3.display\n4.exit")
    choice=int(input())
    if choice==1:
        push()
    elif choice==2:
        pop_element()
    elif choice==3:
        display()
    elif choice==4:
        break
    else:
        print("enter the correct operation")
```

//Program to implement Queue ADT

```
#include <stdio.h>
#define MAX 50
void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
{
    int choice;
    while (1)
{
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
            insert();
            break;
            case 2:
             delete();
            break;
            case 3
            :display();
            break;
            case 4:
            exit(1); default:
            printf("Wrong choice \n");
        } /* End of switch */
    } /* End of while */
} /* End of main() */

void insert()
{
    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty*/
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
} /* End of insert() */
```

```
void delete()
{
   if (front == - 1 || front > rear)
   {
      printf("Queue Underflow \n");
      return ;
   }
   else    {
      printf("Element deleted from queue is : %d\n", queue_array[front]);
      front = front + 1;
   }} /* End of delete() */
 void display()
{
   int i;
   if (front == - 1)
      printf("Queue is empty \n");
   else
   {
      printf("Queue is : \n");
      for (i = front; i <= rear; i++)
         printf("%d ", queue_array[i]);
      printf("\n");
   }} /* End of display() */
```

<u>Output:</u>

```
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit
Enter your choice : 1
Inset the element in queue : 10
1. Insert element to queue
2. Delete element from queue
```

```
3. Display all elements of queue
4. Quit
Enter your choice : 1
Inset the element in queue : 15
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit
Enter your choice : 1
Inset the element in queue : 20
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit
Enter your choice : 1
Inset the element in queue : 30
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit
Enter your choice : 2
Element deleted from queue is : 10
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit
Enter your choice : 3
Queue is :
15 20 30
1. Insert element to queue
2. Delete element from queue
3. Display all elements of queue
4. Quit
Enter your choice : 4
```

#queue using python program

```
def Enqueue():
    if len(queue)==size:
     # check wether the queue is full or not print("Queue is
        Full!!!!")
    else:
        element=input("Enter the element:")
        queue.append(element)
        print(element,"is added to the Queue!")
def dequeue():
    if not queue:# or if len(queue)==0
        print("Queue is Empty!!!")
    else:
        print(queue[0],"element removed!!:",)
        del queue[0] #e=queue.pop(0)
        print()
def display():
    if len(queue)==0:
        print("Queue is Empty!!!")
    else:
        print("Elements of the queue are ")
        for element in queue:
            print(element,end=" ")
        print("\nfront of the queue is ",queue[0])
        print("Rear of the queue is ",queue[-1])
size=int(input("Enter the size of Queue:"))
queue=list()
while(1):

    print("Select the Operation:1.Add 2.Delete 3. Display 4. Quit")
    choice=int(input())
    if choice==1:
        Enqueue()
    elif choice==2:
        dequeue()
    elif choice==3:
        display()
    elif choice==4:
        break
    else:
        print("Invalid Option!!!")
```

WEEK-3

Write C programs to implement list ADT to perform following operations a) Insert an element into a list. b) Delete an element from list c) Search for a key element in list d) count number of nodes in list.

```
//Program to implement list ADT

#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
}*start=NULL,*q,*t;
int main()
{
int ch;
voidinsert_beg();
void insert_end();
int insert_pos();
void display();
void delete_beg();
void delete_end();
int delete_pos();
voidlength();
while(1)
{
printf(&quot;\n\n --- Singly Linked List(SLL) Menu&quot;);
printf(&quot;\n1.Insert\n2.Display\n3.Delete\n4.count\n5.Exit\n\n&quot;);
printf(&quot;Enter your choice(1-5):&quot;);
scanf(&quot;%d&quot;,&amp;ch);
switch(ch)
{
case 1:printf(&quot;\n ---- Insert Menu &quot;);
printf(&quot;\n1.Insertatbeginning\n2.Insert at end\n 3.Insert at specified
position\n4.Exit&quot;);
printf(&quot;\n\nEnter your choice(1-4):&quot;);
scanf(&quot;%d&quot;,&amp;ch);
switch(ch){
case1:
insert_beg();
break;
case2: insert_end();
break;
case 3: insert_pos();
break;
case 4: exit(0);
default: printf(&quot;Wrong Choice!!&quot;); }
break;
case 2: display(); break;
case 3: printf(&quot;\n-----Delete Menu &quot;);
printf(&quot;\n1.Delete from beginning\n2.Delete from end\n 3.Delete from specified
position\n4.Exit&quot;);
printf(&quot;\n\nEnter your choice(1-4):&quot;);
scanf(&quot;%d&quot;,&amp;ch);
switch(ch)
{
case 1: delete_beg();
break;
case 2: delete_end();
break;
case 3: delete_pos();
break;
case 4: exit(0);
default: printf(&quot;Wrong Choice!!&quot;);
}
break;
case4:length();
break;
case 5: exit(0);
default: printf(&quot;Wrong Choice!!&quot;);
}}
return 0;
}
void insert_beg()
{int num;
t=(structnode*)malloc(sizeof(struct node));
printf(&quot;Enter data:&quot;);
scanf(&quot;%d&quot;,&amp;num);
t-&gt;data=num;
if(start==NULL) //If list is empty
{
t-&gt;next=NULL;
start=t;
}else{
t-&gt;next=start;
start=t;
}}
void insert_end(){
int num;
t=(struct node*)malloc(sizeof(struct node));
printf(&quot;Enter data:&quot;);
scanf(&quot;%d&quot;,&amp;num);
t-&gt;data=num;
t-&gt;next=NULL;
if(start==NULL) //If list is empty
{
```

```
start=t;}
else{
q=start;
while(q-&gt;next!=NULL)
q=q-&gt;next;
q-&gt;next=t;}}
int insert_pos(){
int pos,i,num;
if(start==NULL)
{
printf(&quot;List is empty!!&quot;);
return 0;
}
t=(struct node*)malloc(sizeof(struct node));
printf(&quot;Enter data:&quot;);
scanf(&quot;%d&quot;,&amp;num);
printf(&quot;Enterposition to
insert:&quot;); scanf(&quot;%d&quot;,&amp;pos);
t-&gt;data=num;
q=start;
for(i=1;i&lt;pos-1;i++)
{if(q-&gt;next==NULL){
printf(&quot;There are less elements!!&quot;);
return 0;
}
q=q-&gt;next;}
t-&gt;next=q-&gt;next;
q-&gt;next=t;
return 0;}
void length(){
int num=0;
while(start!=NULL){
num+=1;
start=start-&gt;next;
}
printf(&quot;number of nodes in the list is: %d&quot;,num);
}
void display(){
if(start==NULL){
printf(&quot;List is empty!!&quot;);
}
else{
q=start;
printf(&quot;The linked list is:\n&quot;);
while(q!=NULL)
{
printf(&quot;%d-&gt;&quot;,q-&gt;data);
q=q-&gt;next;
}}}
void delete_beg()
```

```
{
if(start==NULL){
printf(&quot;The list is empty!!&quot;);
}
else{

q=start;
start=start-&gt;next;
printf(&quot;Deletedelement is %d&quot;,q-&gt;data);
free(q);
}}
void delete_end()
{
if(start==NULL){
printf(&quot;The list is empty!!&quot;);
}
else{
q=start;
while(q-&gt;next-&gt;next!=NULL)
q=q-&gt;next;
t=q-&gt;next;
q-&gt;next=NULL;
printf(&quot;Deleted element is %d&quot;,t-&gt;data);
free(t); }}
int delete_pos(){
Int pos,i;
if(start==NULL)
{
printf(&quot;List is empty!!&quot;);
return 0; }
printf(&quot;Enterposition to delete:&quot;);
scanf(&quot;%d&quot;,&amp;pos);
q=start;
for(i=1;i&lt;pos-1;i++) {
if(q-&gt;next==NULL) {
printf(&quot;There are less elements!!&quot;);
return 0;
q=q-&gt;next; }
t=q-&gt;next;
q-&gt;next=t-&gt;next;
printf(&quot;Deleted element is %d&quot;,t-&gt;data);
free(t);
return 0; }
```

Output:

```
---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):1
---- Insert Menu ----
1.Insert at beginning 2.Insert at end
```

```
3.Insert at specified position 4.Exit
Enter your choice(1-4):1
Enter data:45

---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):1
---- Insert Menu ----
1.Insert at beginning 2.Insert at end
3.Insert at specified position 4.Exit
Enter your choice(1-4):2
Enter data:67
---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):1
---- Insert Menu ----
1.Insert at beginning 2.Insert at end
3.Insert at specified position 4.Exit
Enter your choice(1-4):3
Enter data:78
Enter position to insert:1
---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):2
The linked list is: 45-&gt;78-&gt;67-&gt;
---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):4 number of nodes in the list is: 3

---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):4 number of nodes in the list is: 0
---- Singly Linked List(SLL) Menu ----
1.Insert 2.Display 3.Delete 4.count 5.Exit
Enter your choice(1-5):5
```

```
# Linked list operations in Python
# Linked list operations in Python


# Create a node
class Node:
    def__init_(self, data):
        self.data = data
        self.next = None
```

```
class LinkedList:

    def___init_(self):
        self.head = None

    # Insert at the beginning
    def insertAtBeginning(self, new_data):
        new_node = Node(new_data)

        new_node.next = self.head
        self.head = new_node

    # Insert after a node
    def insertAfter(self, prev_node, new_data):

        if prev_node is None:
            print("The given previous node must inLinkedList.")
            return

        new_node = Node(new_data)
        new_node.next = prev_node.next
        prev_node.next = new_node

    # Insert at the end
    def insertAtEnd(self, new_data):
        new_node = Node(new_data)

        if self.head is None:
            self.head = new_node
            return

        last = self.head
        while (last.next):
            last = last.next

        last.next = new_node

    # Deleting a node
    def deleteNode(self, position):

        if self.head is None:
            return

        temp = self.head

        if position == 0:
            self.head = temp.next
            temp = None
            return
        # Find the key to be deleted
        for i in range(position - 1):
            temp = temp.next
            if temp is None:
                break

        # If the key is not present
        if temp is None:
            return

        if temp.next is None:
            return

        next = temp.next.next

        temp.next = None

        temp.next = next

    # Search an element
    def search(self, key):

        current = self.head

        while current is not None:
            if current.data == key:
                return True

            current = current.next

        return False


    # Print the linked list
    def printList(self):
        temp = self.head
        while (temp):
            print(str(temp.data) + " ", end="")
            temp = temp.next
    def countNodes(self):
        temp = self.head
        k = 0
        while (temp != None):
            k += 1
            temp = temp.next
        return k

if __name__ == '_main_':

    llist = LinkedList()
    llist.insertAtEnd(1)
    llist.insertAtBeginning(2)
    llist.insertAtBeginning(3)
    llist.insertAtEnd(4)
    llist.insertAfter(llist.head.next, 5)

    print('linked list:')
    llist.printList()


    print("\nAfter deleting an element:")
    llist.deleteNode(3)
    llist.printList()
    print()
    item_to_find = 3
    if llist.search(item_to_find):
        print(str(item_to_find) + " is found")
    else:
        print(str(item_to_find) + " is not found")
    print()
    #count nodes in the list

#number of nodes in the list
    print("No. of nodes: ", llist.countNodes())
```

output:

```
---------
linked list:
3 2 5 1 4
After deleting an element:3 2 5 4
3 is found
No. of nodes: 4
```

WEEK-4

Aim: Write C a n d p y t h o n programs to implement the following using a singlylinked list. a) Stack ADT b) Queue ADT

//Program to implement Stack ADT

```
#include<stdio.h>
#include <stdlib.h>
typedef struct node
{

int data;

struct node *link;

}NODE;

void Push(int);

int pop();
void Display();

NODE *top=NULL; /* Global Declarations */
main()
{/* Main Program
*/ int opn, elem;
do{

printf("\n ### Linked List Implementation of STACK Operations ### \n\n");
printf("\n Press 1-Push, 2-Pop, 3-Display,4-Exit\n");

printf("\n Enter Your option ? ");

scanf("%d",&opn); switch(opn)
{
```

```
case 1:printf("\n\nRead the Element tobe pushed ?");

scanf("%d",&elem);

case 1:Push(elem);

break;

case 2:elem=Pop();
                if(elem != -1)
printf(" Deleted Node(From Top)with the Data: %d\n",elem);

break;

case 3: printf("Linked List Implementation of Stack: Status:\n");

Display();

break;

case 4: printf("\n\n Terminating \n\n"); break;

default: printf("\n\nInvalid Option !!! Try Again !! \n\n");

break;

}

printf("\n\n\n\n Press a Key to Continue . . . ");

}while(opn != 4);

}void Push(int info){

NODE *temp;

temp=(NODE *)malloc(sizeof(NODE));

 if( temp == NULL)

printf(" Out of Memory !! Overflow !!!");

else

{

temp->data=info;

temp->link=top;

 top=temp;

printf(" Node has been inserted at Top(Front) Successfully !!");}}
```

```
int Pop()
{
int info;
 NODE *t;
if( top == NULL)
{ printf(" Underflow!!!");
return -1; }
else{
t=top;
info=top->data;
 top=top->link;
 t->link=NULL;
free(t);
return(info);
}}
void Display()
{
NODE *t;
if( top == NULL)
printf("Empty Stack\n");
else
{
t=top;
 printf("Top->");
 while(t!=NULL)
{
printf("[%d]->",t->data);
t=t->link;
}}}
```

Output:

```
### Linked List Implementation of STACK Operations ### Press 1-Push, 2-Pop, 3-
Display,4-Exit

Your option ? 1

Read the Element tobe pushed ?51

Node has been inserted at Top(Front) Successfully !! Press a Key to Continue . . .

### Linked List Implementation of STACK Operations ### Press 1-Push, 2-Pop, 3-
Display,4-Exit

Your option ? 1

Read the Element tobe pushed ?78

Node has been inserted at Top(Front) Successfully !! Press a Key to Continue . . .

### Linked List Implementation of STACK Operations ### Press 1-Push, 2-Pop, 3-
Display,4-Exit

Your option ? 3

Linked List Implementation of Stack: Status: Top->[78]->[51]->Null

Press a Key to Continue . . .

### Linked List Implementation of STACK Operations ### Press 1-Push, 2-Pop, 3-
Display,4-Exit

Your option ? 2

Deleted Node(From Top)with the Data: 78 Press a Key to Continue . . .

 ### Linked List Implementation of STACK Operations ### Press 1-Push, 2-Pop,
 3- Display,4-Exit

 Your option ? 3

 Linked List Implementation of Stack: Status: Top->[51]-
 
 >Null Press a Key to Continue . . .
```

```
### Linked List Implementation of STACK Operations ### Press 1-Push, 2-Pop,
3- Display,4-Exit
Your option ? 4
```

# Implementation of Stack Using List

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

class Stack:
    def __init__(self):
        self.head = None

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

    def pop(self):
        if self.head is None:
            return None
        else:
            popped = self.head.data
            self.head = self.head.next
            return popped
a_stack = Stack()
```

```
while True:
    print('push <value>')
    print('pop')
    print('quit')
    do = input('What would you like to do? ').split()

    operation = do[0].strip().lower()
    if operation == 'push':
        a_stack.push(int(do[1]))
    elif operation == 'pop':
        popped = a_stack.pop()
        if popped is None:
            print('Stack is empty.')
        else:
            print('Popped value: ', int(popped))
    elif operation == 'quit':
        break
```

output:

```
Case 1:
push <value>
pop
quit
What would you like to do? push 15
push <value>
pop
quit
What would you like to do? push 3
push <value>
pop
quit
What would you like to do? pop
Popped value:  3
```

```
push <value>
pop
quit
What would you like to do? pop
Popped value:  15
push <value>
pop
quit
What would you like to do? pop
Stack is empty.
push <value>
pop
quit
What would you like to do? quit
```

Case 2:

```
push <value>
pop
quit
What would you like to do? pop
Stack is empty.
push <value>
pop
quit
What would you like to do? quit
```

//Program to implement Queue ADT

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int info;
    struct node *next;
}*front,*rear,*temp,temp1;
void enq(int data);
void deq();
void display();
void create();
void queuesize();
int count = 0;
void main()
{
    int  no, ch, e;
    printf("\n 1 - Enque");
    printf("\n 2 - Deque");
    printf("\n 3 - Exit");
    printf("\n 4 - Display");
    printf("\n 5 - Queue size");
    create();
    while (1)
    {       printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1: printf("Enter data : ");
            scanf("%d", &no);
            enq(no);
            break;
        case 2: deq();
```

```
            break;
        case 3: exit(0);
         case 4: display();
            break;
         case 5: queuesize();
            break;
         default: printf("Wrong choice, Please enter correct choice  ");
            break;
         } }} /* Create an empty queue */
void create()
{
    front = rear = NULL;
} /* Returns queue size */
void queuesize(){
    printf("\n Queue size : %d", count);
}/* Enqueing the queue */
void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->next = NULL;
        rear->info = data;
        front = rear;
    }
    else    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->next = temp;
        temp->info = data;
        temp->next = NULL;
        rear = temp;
    }
    count++;}
```

```
/* Displaying the queue elements */
void display()
{
   temp1= front;
    if ((temp1 == NULL) && (rear == NULL))
   {
      printf("Queue is empty");
      return;
   }
   while (temp1!= NULL)    {
      printf("%d ", temp1->info);
      temp1 = temp1->next;    }}
/* Dequeing the queue */
void deq()
{
   temp1 = front;
   if (temp1 == NULL)
   {
      printf("\n Error: Trying to display elements from empty queue");
      return;
   }
   else       {
         front = front->next;
         printf("\n Dequed value : %d", temp1->info);
         free(temp1);
      }
      count--;  }
/* Display if queue is empty or not */
void empty()
{
    if ((front == NULL) && (rear == NULL))
      printf("\n Queue empty");
   else
```

```
    printf("Queue not empty");
}
```

Output:

```
1-Enque
2 - Deque
3 - Exit
4 - Display
5 - Queue size
Enter choice : 1
Enter data : 14
Enter choice : 1
Enter data : 85
Enter choice : 1
Enter data : 38
Enter choice : 4
14 85 38
Enter choice : 5
Queue size : 3
Enter choice : 2
Dequed value : 14
Enter choice : 4
85 38
Enter choice : 5
Queue size : 2
Enter choice : 3
```

//Program to implement Queue ADT Using Python

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

class Queue:
    def init (self):
        self.head = None
        self.last = None

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

    def dequeue(self):
        if self.head is None:
            return None
        else:
            to_return = self.head.data
            self.head = self.head.next
            return to_return

a_queue = Queue()
while True:
    print('enqueue <value>')
    print('dequeue')
    print('quit')
    do = input('What would you like to do? ').split()

    operation = do[0].strip().lower()
    if operation == 'enqueue':
        a_queue.enqueue(int(do[1]))
    elif operation == 'dequeue':
        dequeued = a_queue.dequeue()
        if dequeued is None:
            print('Queue is empty.')
        else:
            print('Dequeued element: ', int(dequeued))
    elif operation == 'quit':
        break
```

OUTPUT:

```
Case 1:
enqueue <value>
dequeue
quit
What would you like to do? enqueue 3
enqueue <value>
dequeue
quit
What would you like to do? enqueue 4
enqueue <value>
dequeue

quit
What would you like to do? dequeue
Dequeued element:  3
enqueue <value>
dequeue
quit
What would you like to do? dequeue
Dequeued element:  4
enqueue <value>
dequeue
quit
What would you like to do? dequeue
```

```
Queue is empty.
enqueue <value>
dequeue
quit
What would you like to do? quit
```

Case 2:

```
enqueue <value>
dequeue
quit
What would you like to do? dequeue
Queue is empty.
enqueue <value>
dequeue
quit
What would you like to do? enqueue 5
enqueue <value>
dequeue
quit
What would you like to do? dequeue
Dequeued element:  5
enqueue <value>
dequeue
quit

What would you like to do? quit
```

WEEK-5

Aim: Write C & Python programs to implement the deque (double endedqueue) ADT using a doubly linked list and an array

//C Program to implement deque ADT using array.

```
 #include<stdio.h>
 #include<stdlib.h>
 # define SIZE 6
int a[SIZE],data,front=-1,rear=-1,count;
 void enqueuerear()
 {

 if(front==-1){

 front++;

 rear++;
 printf("\n enter element to insert");
 scanf("%d",&a[rear]);
 count++; }

 else  if(rear>=SIZE-1)

 {

printf("\nInsertion is not possible,overflow!!!\n");
 return;
 }else {
 printf("\n enter element to insert");

 scanf("%d",&a[++rear]);
 count++;}}

 void enqueuefront() {

 int  i;  if(front==-1)  {

 front++;

  rear++;

printf("\n enter element to insert");
 scanf("%d",&a[rear]);

 count++;
 }

 else if(rear>=SIZE-1)  {

printf("\nInsertion is not possible,overflow!!!!\n");
 return;
 }else {
 for(i=count;i>0;i--)
 a[i]=a[i-1];
printf("\n enter element to insert");
 scanf("%d",&a[i]);
count++;
 rear++;
 }}

 void dequeuerear() {
 if(front==-1)

 {

 printf("Deletion is not possible:Dequeue is empty\n");
 return;
 }

 else
```

```
printf("\nThe deleted element is %d ",a[rear]);
if(front==rear)
front=rear=-1;
else
rear=rear-1; }

void dequeuefront() {

if(front==-1)

{

printf("Deletion is not possible:: Dequeue is empty\n");
return;
}

else

printf("The deleted element is %d ",a[front]);
if(front==rear)
front=rear=-1;
else
front=front+1;
}

void display() {

int i;

if(front==-1){

printf("queue empty\n");

return;
}
for(i=front;i<=rear;i++)
printf("%d \t",a[i]);
}

int main(){

int ch;
```

```
printf("1.enquerear\n2-enquefront\n3-dequerear\n4-dequefront\n5-display\n6-
exit\n");

while(1) {

printf("enter yr choice");

scanf("%d",&ch);

switch(ch) {

case 1:enqueuerear();

break;

case2:enqueuefront();

break;

case 3:dequeuerear();

break;

case 4:dequeuefront();

break;

 case 5:display();

break;
case 6:exit(0);

}}}
```

Output:

```
1. enquerear 2-enquefront 3-dequerear 4-dequefront 5-display 6-exit
enter yr choice1


enter element to insert23
enter yr choice2
enter element to insert34
enter yr choice5 34 23
enter yr choice3

The deleted element is 23

enter yr choice5 34
enter yr choice 6
```

//C Program to implement deque ADT using dll

```
#iinclude<stdio.h>
#include<stdlib.h>
struct node
{

int data;

struct node*left,*right;

};

struct    node *front=NULL,*rear=NULL,*cur,*temp;
void insertrear()
{

cur=(struct node*)malloc(sizeof(struct node));

 printf("\n enter data to insert ");

scanf("%d",&cur->data);

 cur->right=NULL;
if(rear==NULL&&front==NULL)
rear=front=cur;
else{

rear->right=cur;

cur->left=rear;
rear=cur;}}

void insertfront() {

cur=(structnode*)malloc(sizeof(struct node));

printf("\n enter data to insert ");

 scanf("%d",&cur->data);
cur->right=NULL;
if(rear==NULL&&front==NULL)
rear=front=cur;
else{

cur->right=front;

front->left=cur;

front=cur;}}

void delfront(){

if(front==NULL)

printf("\ndeQueueis empty\n");
else if(front==rear)
{

printf("\nDeleted data is %d ",front->data);
front=rear=NULL;
}

Else {

temp=front;

printf("\nDeleted data is %d ",temp->data);
front=front->right;
front->left=NULL;
 free(temp); }}

void delrear(){
```

```
if(front==NULL)

printf("\ndeQueue is empty\n");

else if(front==rear){

printf("\nDeleted data is %d ",rear->data);
front=rear=NULL;
}

else{

temp=rear;

printf("\nDeleted data is %d",temp->data);

if(front==rear)
front=rear=NULL;

rear=rear->left;

rear->right=NULL;
free(temp); }}

void display()

{

if(front==NULL)

printf("\ndeQueue is empty\n");

else{

temp=front;

while(temp!=NULL)
{

printf("<-%d ->",temp->data);

temp=temp->right; }}}

int main()
```

```
{
int ch;
printf("\n1.InsertFront\n2.Deletefront\n3.Insertrear\n4.DeleteRear\n5.Display\n
6.Exit);
while(1){
printf("\nEnter your choice ");
scanf("%d",&ch);
switch(ch){
case 1: insertfront(); break;
case 2: delfront(); break;
case 3: insertrear(); break;
case 4: delrear(); break;
case 5: display(); break;
 case 6: exit(0);
}//end of switch
}/}
```

Output:

```
1.Insert Front 2.Delete front 3.Insert rear 4.Delete Rear 5.Display 6.Exit
Enter your choice 1
enter data to insert 67
Enter your choice 1
enter data to insert 78
Enter your choice 3
```

```
enter data to insert 45

Enter your choice 5

<-78 -><-67 -><-45 ->

Enter your choice 2
Deleted data is 78
Enter your choice 6
```

Python program for Deque(Double Ended Queue)

```
import collections
 data = collections.deque([7, 8, 9, ])
data.extend([1, 2, 3])
print("The deque at right side: ")
print(data)
data.extendleft([4, 5, 6])
print("The deque at left side: ")
print(data)
data.rotate(-4)
print("After rotation operation the deque is : ")
print(data)
data.reverse()
print("The reversing deque is : ")
print(data)
```

output:

```
The deque at right side:
deque([7, 8, 9, 1, 2, 3])
The deque at left side:
deque([6, 5, 4, 7, 8, 9, 1, 2, 3])
After rotation operation the deque is :
deque([8, 9, 1, 2, 3, 6, 5, 4, 7])
The reversing deque is :
deque([7, 4, 5, 6, 3, 2, 1, 9, 8])
```

WEEK-6

Aim: Write a C program to perform the following operations: a) Insert an element into a binary search tree. b) Delete an element from a binary search tree c) Search for a key element in a binary search tree..

//C Program to implement Binary search tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int info;
struct    node*left;
struct node*right;
};

typedef struct node BST;

BST *LOC, *PAR;
void search(BST *root, int item)

{

BST *save,*ptr;

if (root == NULL){

LOC = NULL;

PAR=NULL;}

if (item == root -> info){

LOC = root;
PAR = NULL;
 return;}
 if (item < root->info){
 save = root;
 ptr = root->left;}
 else{
 save = root;
 ptr = root -> right;
 }while( ptr != NULL){
if (ptr -> info == item){
LOC = ptr;
PAR = save;
 return;
 }if(item < ptr->info){
 save = ptr;
 ptr = ptr->left;}
 else{
 save = ptr;
 ptr = ptr->right;
 }}
 LOC = NULL;
PAR = save;
 return;}
 struct node* findmin(struct node*r)
```

```
{
if (r == NULL)
return NULL;
else if (r->left!=NULL)
return findmin(r->left);
else if (r->left == NULL)
return r;
}
struct node*insert(struct node*r, int x){
if (r == NULL){
r = (struct node*)malloc(sizeof(struct node));
r->info = x;
r->left = r->right = NULL;
return r;
}
else if (x < r->info)
r->left = insert(r->left, x);
else if (x > r->info)
r->right = insert(r->right, x);
return r;
}
struct node* del(struct node*r, int x){
struct node *t;
if(r == NULL)
printf("\nElement not found");
```

```
 else if (x < r->info)
 r->left = del(r->left, x);
 else if (x > r->info)
 r->right = del(r->right, x);

 else if ((r->left != NULL) && (r->right != NULL))

 {

t = findmin(r->right);

 r->info = t->info;
 r->right = del(r->right, r->info);}

 else{

 t = r;

if (r->left == NULL)

r = r->right;
 else if (r->right == NULL)
 r = r->left;
 free(t);}

 return r; }

 int main()

 {

struct node* root = NULL;

 int x, c = 1, z;

int element;
 char ch;
printf("\nEnter an element: ");
 scanf("%d", &x);
 root = insert(root, x);
 printf("\nDo you want to enter another element :y or n");
 scanf(" %c",&ch);
 while (ch == 'y'){
 printf("\nEnter an element:");
 scanf("%d", &x);
 root = insert(root,x);
 printf("\nPress y or n to insert another element: y or n: ");
 scanf(" %c", &ch);
 }
 while(1){
 printf("\n1 Insert an element ");
 printf("\n2 Delete an element");
 printf("\n3 Search for an element ");
 printf("\n4 Exit ");
printf("\nEnter your choice: ");
 scanf("%d", &c);
 switch(c){
 case 1:printf("\nEnter the item:");
 scanf("%d", &z);
 root = insert(root,z);
 break;
 case 2:printf("\nEnter the info to be deleted:");
  scanf("%d", &z);
 root = del(root, z); break;
 case 3:printf("\nEnter element to be searched: ");
 scanf("%d", &element);
search(root, element);
 if(LOC != NULL)
printf("\n%d Found in Binary Search Tree !!\n",element);
 else
printf("\nIt is not present in Binary Search Tree\n");
 break;
 case4: printf("\nExiting...");
  return;
 default:printf("Enter a valid choice: ");

 }}

 return 0;}
```

Output:

```
Enter an element: 18

Do you want to enter another element :y or ny Enter an element:56
Press y or n to insert another element: y or n: n
1 Insert an element 2 Delete anelement
3 Search for an element 4 Exit
Enter your choice: 1 Enter the item:36
1 Insert an element 2 Delete anelement
3 Search for an element 4 Exit
Enter your choice: 3
Enter element to be searched: 18 18
Found in Binary Search Tree !!

1 Insert an element 2 Delete anelement
3 Search for an element 4 Exit
Enter your choice: 4 Exiting...
```

WEEK-7

Aim: Write C and Python programs that use recursive functions to traverse thegiven binary tree in a)Preorder b) inorder and c) postorder.

//Program to traverse the given binary tree

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct BST
{char data;
struct BST *left,*right;
}node;
struct BST* root=NULL,*temp,*cur;
void create()
{
char c[10];
temp=root;
cur=(struct BST*)malloc(sizeof(struct BST));
printf("\n enter character:\n");
fflush(stdin);
scanf("%s",c);
cur->data=c[0];
cur->left=NULL;
```

```
cur->right=NULL;
if(temp==NULL)
root=cur;
else{
while(temp!=NUL
L){
if((cur->data)<(temp->data)){
if(temp->left==NULL) {
temp->left=cur;
return;
}
else{
temp=temp->left; }
else
if(temp->right==NULL)
{
temp->right=cur;
return;
}}}}//create
void preorder(struct BST *temp)
{
if(temp!=NULL){
printf("\t%c",temp->data);
preorder(temp->left);
preorder(temp->right);}}
void Inorder(struct BST *temp) {
```

```
if(temp!=NULL) {

 Inorder(temp->left);

  printf("\t%c",temp->data);

  Inorder(temp->right);  }}

void postorder(struct BST *temp)

{

if(temp!=NULL) {

postorder(temp->left);

postorder(temp->right);

printf("\t%c",temp->data); }}

int main() {

int ch;

printf("\nmenuoptions\n");
printf("1.Create\n2.Preorder\n3.Inorder\n4..Postorder\n5.exit\n");
while(1){

printf("\nenterur choice");

scanf("%d",&ch);

switch(ch) {

case1:create(); break;
case 2:printf("Preorder Traversal\n");
preorder(root);break;

case3:printf("Inorder Traversal\n");
Inorder(root);break;

case4:printf("Postorder  Traversal\n");

postorder(root);break;
case5:exit(0);
default:printf("invalid choice\n");
}//switch
}//while
}//main
```

Output:

```
menu options 1.Create 2.Preorder 3.Inorder 4.Postorder 5.exit
enter ur choice 1
enter  character:  r
enter  ur  choice  1
enter  character:  t
enter  ur  choice  1
enter character: y
enter ur choice 4
Postorder   Traversal   y   t    r
enter ur choice 5
```

# Tree traversal in Python

```
class Node:
    def_init (self, item):
self.left = None
self.right = None
self.val = item
def inorder(root):
    if root:
        # Traverse left
inorder(root.left)
        # Traverse root
        print(str(root.val) + "->", end='')
        # Traverse right
inorder(root.right)
def postorder(root):
    if root:
        # Traverse left
postorder(root.left)
         # Traverse right
  postorder(root.right)
         # Traverse root
        print(str(root.val) + "->", end='')
def preorder(root):

    if root:
        # Traverse root
        print(str(root.val) + "->", end='')
        # Traverse left
preorder(root.left)
        # Traverse right
preorder(root.right)
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)

print("Inorder traversal ")
inorder(root)
print("\nPreorder traversal ")
preorder(root)

print("\nPostorder traversal ")
postorder(root)
```

WEEK-8

Aim: Write a C program to perform the following operations

a) Insertion into a B-tree b) Deletion from a B-tree

//C Program to implement B-tree

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 4
#define    MIN    2
struct btreeNode {
int val[MAX + 1], count;
struct btreeNode *link[MAX + 1];

};

struct btreeNode *root;

/* creating new node */

struct btreeNode * createNode(int val, struct btreeNode *child)

{ struct btreeNode *newNode;

newNode = (struct btreeNode *)malloc(sizeof(struct btreeNode));
newNode->val[1] = val;
newNode->count = 1;
newNode->link[0] = root;
newNode->link[1] = child;
return newNode;
}
```

```
/* Places the value in appropriate position */

void addValToNode(int val, int pos, struct btreeNode *node, struct btreeNode *child)
{
int j = node->count;
while (j > pos) {
node->val[j + 1] = node->val[j];
node->link[j + 1] = node->link[j];
j--; }

node->val[j + 1] = val;

node->link[j + 1] = child;

node->count++;}

/* split the node */

void splitNode (int val, int *pval, int pos, struct btreeNode *node, struct btreeNode
*child, struct btreeNode **newNode)
{
int median, j;
if(POS<MIN)

 median = MIN + 1;

  else
median = MIN;
*newNode = (struct btreeNode *)malloc(sizeof(struct btreeNode));

 j = median + 1;
while (j <= MAX) {
(*newNode)->val[j - median] = node->val[j];

(*newNode)->link[j - median] = node->link[j];
```

```
j++; }

node->count = median;

(*newNode)->count = MAX - median;

 if (pos <= MIN) {
addValToNode(val, pos, node, child);

} else {

addValToNode(val, pos - median, *newNode, child);

}

*pval = node->val[node->count];

(*newNode)->link[0]= node->link[node->count];
node->count--;  }
/* sets the value val in the node */

int setValueInNode(int val, int *pval,

struct btreeNode *node, struct btreeNode **child) {

int pos;

if (!node) {

*pval = val;

*child = NULL;

return 1;}

if (val < node->val[1])

{ pos = 0;

} else {

for (pos = node->count;(val < node->val[pos] && pos > 1); pos--);
```

```
if (val == node->val[pos])
{
printf("Duplicatesnot allowed\n");
return 0; }}
if (setValueInNode(val, pval, node->link[pos], child))
{if (node->count < MAX) { addValToNode(*pval, pos, node, *child);
} else {
splitNode(*pval, pval, pos, node, *child, child);
return 1; }}
return 0; }
/* insert val in B-Tree */ void insertion(int val)
{ int flag, i;
struct btreeNode *child;
flag = setValueInNode(val, &i, root, &child);
if (flag)
root = createNode(i, child);  }
/* copysuccessor for the value to be deleted */
void copySuccessor(struct btreeNode *myNode, int pos)
 { struct btreeNode *dummy;
 dummy = myNode->link[pos];
 for (;dummy->link[0] != NULL;)
 dummy = dummy->link[0];
myNode->val[pos] = dummy->val[1]; }
/* removes the value from the given node and rearrange values */
void removeVal(struct btreeNode *myNode, int pos) {
```

```
int i = pos + 1;

while (i <= myNode->count)

 { myNode->val[i - 1] = myNode->val[i];

 myNode->link[i - 1] = myNode->link[i];

i++; }

myNode->count--;

}/* shifts value from parent to right child */

void doRightShift(struct btreeNode *myNode, int pos) {

 struct btreeNode *x = myNode->link[pos];

int j = x->count;

while (j > 0) {

x->val[j + 1] = x->val[j];

x->link[j + 1] = x->link[j];  }

x->val[1] = myNode->val[pos];

x->link[1] = x->link[0];

 x->count++;

x = myNode->link[pos - 1];

myNode->val[pos] = x->val[x->count];

myNode->link[pos] = x->link[x->count];

x->count--;

return; }

/* shifts value from parent to left child */

void doLeftShift(struct btreeNode *myNode, int pos)

{ int j = 1;
```

```
struct btreeNode *x = myNode->link[pos - 1];
x->count++;
x->val[x->count] = myNode->val[pos];
x->link[x->count] = myNode->link[pos]->link[0];
x = myNode->link[pos];
myNode->val[pos] = x->val[1];
x->link[0] = x->link[1];
x->count--;
while (j <= x->count) {
x->val[j] = x->val[j + 1];
x->link[j] = x->link[j + 1];
j++;  }
return;}
/* merge nodes */
void mergeNodes(struct btreeNode *myNode, int pos)
{ int  j = 1;
struct btreeNode *x1 = myNode->link[pos], *x2 = myNode->link[pos - 1];
x2->count++;
x2->val[x2->count]  =  myNode->val[pos];
x2->link[x2->count]  =  myNode->link[0];
while (j <= x1->count) {
x2->count++;
x2->val[x2->count] = x1->val[j];
x2->link[x2->count] = x1->link[j];
j++;  }
j = pos;
```

```
while (j < myNode->count) {

myNode->val[j] = myNode->val[j + 1];

myNode->link[j] = myNode->link[j + 1];

j++;}

myNode->count--;

free(x1);

}

/* adjusts the given node */
void adjustNode(struct btreeNode *myNode, int pos)

{ if (!pos)

{

if (myNode->link[1]->count > MIN)

{ doLeftShift(myNode, 1);

} else { mergeNodes(myNode, 1);

}

} else {

if (myNode->count != pos) {

if(myNode->link[pos - 1]->count > MIN)

{ doRightShift(myNode, pos);

} else {

if (myNode->link[pos + 1]->count > MIN)

{ doLeftShift(myNode, pos + 1);

} else { mergeNodes(myNode, pos);
}}} else {
if (myNode->link[pos - 1]->count > MIN)
doRightShift(myNode, pos);
else
mergeNodes(myNode, pos);
}}}
/* delete val from the node */
int delValFromNode(int val, struct btreeNode *myNode)
{ int pos, flag = 0;
if (myNode) {
if (val < myNode->val[1])
{ pos = 0;
flag = 0;
} else {
for (pos = myNode->count;(val < myNode->val[pos] && pos > 1); pos--);
if (val == myNode->val[pos]) {
flag = 1;
} else { flag = 0;
}}if (flag) {
if (myNode->link[pos - 1])
{ copySuccessor(myNode, pos);
flag = delValFromNode(myNode->val[pos], myNode->link[pos]);
if (flag == 0) {
printf("Given data is not present in B-Tree\n");
}} else { removeVal(myNode, pos);
}} else {
flag = delValFromNode(val, myNode->link[pos]);
}
if (myNode->link[pos]) {
if (myNode->link[pos]->count < MIN) adjustNode(myNode, pos); }}
return flag;}
/* delete val from B-tree */
void deletion(int val, struct btreeNode *myNode) {
 struct btreeNode *tmp;
if (!delValFromNode(val, myNode))
{
printf("Given value is not present in B-Tree\n");
return;
} else {
if (myNode->count == 0)
{ tmp = myNode;
myNode = myNode->link[0];
free(tmp);
}}
root = myNode;
return;}
/* B-Tree Traversal */
void traversal(struct btreeNode *myNode)
```

```
 { int i;

 if (myNode) {

 for (i = 0; i < myNode->count; i++) {

 traversal(myNode->link[i]);

 printf("%d ", myNode->val[i + 1]);
 }
 traversal(myNode->link[i]);

 }}

 int main()
 { int val, ch;
 while (1) {
printf("1. Insertion\t2. Deletion\n");
 printf(3. Traversal\n");
printf("4.    Exit\nEnter    your    choice:");
 scanf("%d", &ch);
 switch (ch) {

case 1: printf("Enter your input:");
 scanf("%d", &val);
 insertion(val); break;
 case 2: printf("Enter the element to delete:");
 scanf("%d", &val);
deletion(val, root);
 break;
case 3: traversal(root);
 break;
 case 4: exit(0);
 default: printf("U have entered wrong option!!\n");
 break;
 }
 printf("\n");
 }}
```

Output:

```
1. Insertion 2. Deletion
3.     Traversal
4.     Exit
Enter your choice:1
Enter your input:21

1. Insertion 2. Deletion
3.     Traversal
4.     Exit
Enter your choice:1
 Enter your input:67

1. Insertion 2. Deletion
3.     Traversal
4.     Exit
Enter your choice:3 21 67
1. Insertion 2. Deletion
3.     Traversal
```

```
4.    Exit

Enter your choice:2

Enter the element to delete:67


1. Insertion 2. Deletion

3.    Traversal

4.    Exit

Enter your choice:3 21

1. Insertion 2. Deletion

3.    Traversal

4.    Exit

Enter your choice:4
```

WEEK-9

Aim: Write a C program to perform the following operations

a) Insertion into an AVL-tree b) Deletion from an AVL-tree.

//C Program to implement AVL-tree

```
#include <stdio.h>
#include <stdlib.h>
struct AVLTree_Node
{
int data, bfactor;
struct AVLTree_Node *link[2];
};
struct AVLTree_Node *root = NULL;
struct AVLTree_Node * createNode(int data)
{ struct AVLTree_Node *newnode;
newnode   = (struct    AVLTree_Node    *)malloc(sizeof (struct AVLTree_Node));
newnode->data = data;
newnode->bfactor = 0;
newnode->link[0] = newnode->link[1] = NULL;
return newnode;}
void insertion (int data) {
struct AVLTree_Node *bf, *parent_bf, *subtree, *temp;
struct AVLTree_Node *current, *parent, *newnode, *ptr;
```

```
int res = 0, link_dir[32], i = 0;
 if (!root) {
 root = createNode(data);
 return; }
 bf = parent_bf = root;
 /* find the location for inserting the new node*/
 for (current = root; current != NULL; ptr = current, current = current->link[res])
  { if (data == current->data) {
  printf("Cannot insert duplicates!!\n");
  return; }
res = (data > current->data) ? 1 : 0;
 parent = current;
 if (current->bfactor != 0)
 {
  bf = current;
  parent_bf = ptr;
  i = 0; }
 link_dir[i++] = res; }
 /* create the new node */ newnode = createNode(data);
 parent->link[res] = newnode;
  res = link_dir[i = 0];
 /* updating the height balance after insertion */
 for (current = bf; current != newnode; res = link_dir[++i])
 { if (res == 0)
 current->bfactor--;

 else

 current->bfactor++;
 current = current->link[res]; }
 /* right sub-tree */

 if (bf->bfactor == 2)
{ printf("bfactor = 2\n");
 temp = bf->link[1];
 if (temp->bfactor == 1)

 { subtree = temp;

bf->link[1] = temp->link[0];

 temp->link[0] = bf;
 temp->bfactor = bf->bfactor = 0;

 } else {

 subtree = temp->link[0];

temp->link[0] = subtree->link[1];

subtree->link[1] = temp;
 bf->link[1] = subtree->link[0];
 subtree->link[0] = bf;
 /* update balance factors */

 if (subtree->bfactor == -1)
 { bf->bfactor = 0;

 temp->bfactor = 1;

 } else if (subtree->bfactor == 0)
 { bf->bfactor = 0;

 temp->bfactor = 0;
```

```
} else if (subtree->bfactor == 1)
{ bf->bfactor = -1;

temp->bfactor = 0;
}subtree->bfactor = 0;
}

/* left sub-tree */

} else if (bf->bfactor == -2)

{ temp = bf->link[0];

if (temp->bfactor == -1)

{ subtree = temp;

bf->link[0] = temp->link[1];

temp->link[1] = bf;
temp->bfactor = bf->bfactor = 0;

} else {

subtree = temp->link[1];

temp->link[1] = subtree->link[0];
subtree->link[0] = temp;
bf->link[0] = subtree->link[1];
subtree->link[1] = bf;
/* update balance factors */
if (subtree->bfactor == -1)

{ bf->bfactor = 1;

temp->bfactor = 0;
}
else if (subtree->bfactor == 0)
{ bf->bfactor = 0;
```

```
 temp->bfactor = 0;
} else if (subtree->bfactor == 1)
{ bf->bfactor = 0;
temp->bfactor = -1;
}
subtree->bfactor = 0;
}} else {
 return;}
if (bf == root)
{ root = subtree;
return;
}
if (bf != parent_bf->link[0])
{ parent_bf->link[1] = subtree;
} else {
parent_bf->link[0] = subtree;
}
return;}
void deletion(int data) {
int link_dir[32], res = 0, i = 0, j = 0, index = 0;
struct AVLTree_Node *ptr[32], *current, *temp, *x, *y, *z;
current = root;
if (!root) {
printf("Tree not present\n");
return;}
if ((root->data == data) && (root->link[0] == NULL) && (root->link[1] == NULL))
{
```

```
free(root);
root = NULL;
return;
}
/* search the node to delete */
while (current != NULL) {
if (current->data == data)
break;
res = data > current->data ? 1 : 0;
 link_dir[i] = res;
ptr[i++] = current;
current = current->link[res]; }
if (!current) {
printf("Given data is not present!!\n");

return;}

index = link_dir[i - 1];

 temp = current->link[1];
/* delete the node from the AVL tree - similar to BST deletion */
if (current->link[1] == NULL) {
if (i == 0) {
temp = current->link[0];
free(current);
root = temp;
return;
} else {

ptr[i - 1]->link[index] = current->link[0];  }

} else if (temp->link[0] == NULL)

 { temp->link[0] = current->link[0];

  temp->bfactor = current->bfactor;
 if (i > 0) {
ptr[i-1]->link[index] = temp;

} else {

root = temp; }

link_dir[i] = 1;

ptr[i++] = temp;
} else {
/* delete node with two children */

j = i++;

 while (1) {
link_dir[i] = 0;
ptr[i++] = temp;
x = temp->link[0];

if (x->link[0] == NULL) break;
temp = x;
}
x->link[0] = current->link[0];

temp->link[0] = x->link[1];
x->link[1]  =  current->link[1];
x->bfactor  = current->bfactor;
if (j > 0) {
ptr[j - 1]->link[index] = x;

} else { root = x; }

link_dir[j] = 1;

ptr[j] = x; }

free(current);
```

```
for (i = i - 1; i >= 0; i = i--)

{ x = ptr[i];

if (link_dir[i] == 0)

{ x->bfactor++;

if (x->bfactor == 1)

{ break;

} else if (x->bfactor == 2)

{ y = x->link[1];

if (y->bfactor == -1) {

/* double rotation - (SR right + SR left) */

z = y->link[0];

 y->link[0] = z->link[1];

z->link[1] = y;
x->link[1] = z->link[0];
 z->link[0] = x;

/* update balance factors */

 if (z->bfactor == -1) {

 x->bfactor = 0;
y->bfactor = 1;
} else if (z->bfactor == 0)

{

x->bfactor = 0;

y->bfactor = 0;

} else if (z->bfactor == 1)
```

```
 {
 x->bfactor = -1;
 y->bfactor = 0;
 }
 z->bfactor = 0;
 if (i > 0) {
index = link_dir[i - 1];
 ptr[i - 1]->link[index] = z;
 } else { root = z;
 }
 } else {
/* single rotation left */
x->link[1] = y->link[0];
 y->link[0] = x;
 if (i > 0) {
index = link_dir[i - 1];
 ptr[i - 1]->link[index] = y;
 } else {
  root = y;
 }
 /* update balance factors */
 if (y->bfactor == 0) {
  x->bfactor = 1;
 y->bfactor = -1;
 break;
} else {

x->bfactor = 0;

y->bfactor = 0; }}}} else {

x->bfactor--;

if (x->bfactor == -1){

break;

} else if (x->bfactor == -2) {

y = x->link[0];

if (y->bfactor == 1) {

/* double rotation - (SR right + SR left) */

 z = y->link[1];

y->link[1] = z->link[0];
z->link[0] = y;

x->link[0]=z->link[1];

z->link[1] = x;
/* update balance factors */
 if (z->bfactor == -1) {
 x->bfactor = 1;
y->bfactor = 0;

} else if (z->bfactor == 0)

{

x->bfactor = 0;

y->bfactor = 0;

} else if (z->bfactor == 1)
```

```
 {x->bfactor = 0;

 y->bfactor = -1;}

z->bfactor = 0;
 if (i > 0) {

 index = link_dir[i - 1];

  ptr[i - 1]->link[index] = z;
 } else { root = z;

 }} else {

 /* single rotation right */

 x->link[0]  =  y->link[1];

 y->link[1] = x;
 if (i <= 0) {

 root = y;

 } else {

 index = link_dir[i - 1];

 ptr[i - 1]->link[index] = y;
 }
 /* update balance factors */

 if (y->bfactor == 0) {

 x->bfactor = -1;

 y->bfactor = 1;
 break;

 } else {

 x->bfactor = 0;

 y->bfactor = 0;}}}}}}
 int main()

 {

 int key, ch;

 while (1) {

printf("1. Insertion\t2. Deletion\n");

 printf("3. Exit\nEnter your choice:");

scanf("%d", &ch);

switch (ch) {

case 1:

 printf("Enter the key value:");

 scanf("%d",&key);

 insertion(key);

 break;

case 2: printf("Enter the key value to delete:");

 scanf("%d", &key);

 deletion(key);

  break;

 case 3:exit(0);

 default:printf("Wrong Option!!\n");

 break;

 }

 printf("\n"); }

 return 0; }
```

Output:

```
1. Insertion 2. Deletion 3. Exit

Enter your choice: 1

 Enter the key value: 24

1. Insertion 2. Deletion
3. Exit

Enter your choice: 1

Enter the key value: 45

1. Insertion 2. Deletion

3. Exit

Enter your choice: 1

 Enter the key value: 67

 bfactor = 2

1. Insertion 2. Deletion

3. Exit

Enter your choice: 1 Enter the key value: 78

1. Insertion 2. Deletion

3. Exit

Enter your choice: 2

Enter the key value to delete: 67
1. Insertion 2. Deletion

3. Exit

Enter your choice:3
```

WEEK-10

Aim: Write a C &Python program to implement hash table and perform the following operations

b) Inserting a key-value pair b) Deleting a key-value pair

•

// C program to implement ADT using hashing

```
#include<stdio.h>
 #include<stdlib.h>
 #include<string.h>
 int b;
 int hsearch(int key,int d,int *ht,int *empty)

 {

int i=key%(d);
 int j=i;
 int c=0;
 do
 {

if(empty[j]||(*(ht+j)==key))
 return j;
 c++;

 j=(i+c)%(d);

 }while(j!=i);

  return 0;
 }
 int search(int key,int d,int *ht,int *empty)
```

```
{

b=hsearch(key,d,ht,empty);

printf("%d",b);

if(empty[b]==1)

return -1;

else if(b==0)

return 1;

 else

return b;}

/*insertion operation*/

void insert(int key,int d,int *ht,int *empty) {

b=hsearch(key,d,ht,empty);
if(empty[b])
{

empty[b]=0;

*(ht+b)=key;

printf("elements is inserted\n"); }}/*deletion operation*/

void delete(int key,int d,int *ht,int *empty) {

int b=hsearch(key,d,ht,empty);

*(ht+b)=0;
empty[b]=1;
printf("element is deleted\n"); }

void display(int d,int *ht,int *empty) {

int i;

printf("hash table elements are\n");
```

```
 for(i=0;i<d;i++) {

if(empty[i])

printf(" 0");

else

printf("%5d",*(ht+i)); }
printf("\n"); }/*main program*/

 void main(){

int choice=1;

int key;
int d,i,s;
int *empty,*ht;
printf("enter the hash table size:");
scanf("%d",&d);
ht=(int *)malloc(d *sizeof(int));
empty=(int *)malloc(d *sizeof(int));
for(i=0;i<d;i++)
empty[i]=1;
while(1) {
printf("\n");
printf("\n LINEAR PROBING");
printf("\n 1:insert hash table:");
printf("\n 2:delete hash table");
printf("\n 3:search hash table");
printf("\n 4:display hash table");
printf("\n 5:exit");
printf("enter your choice");
scanf("%d",&choice);
switch(choice) {

case 1:printf("enter the elemants:");
```

```
 scanf("%d",&key);
 insert(key,d,ht,empty);
break;
case 2: printf("enter to remove from hash table:");
scanf("%d",&key);
delete(key,d,ht,empty);
break;
case 3: printf("enter the search elements:");
scanf("%d",&key);
s=search(key,d,ht,empty);
if(s==-1||s==0)
 printf("not found\n");
 else
printf("element found at index%d",hsearch(key,d,ht,empty));
break;
case4: display(d,ht,empty);
 break;
case 5: exit(0);  }
}return;}
```

Output:

```
enter the hash table size:4
LINEAR PROBING
1:insert hash table: 2:delete hash table 3:search hash table 4:display hash table
5:exitenter your choice1 enter the elemants:67 elements is inserted


LINEAR PROBING

1:insert hash table: 2:delete hash table 3:search hash table 4:display hash table
5:exitenter your choice1 enter the elemants:78 elements is inserted
```

```
LINEAR PROBING

1:insert hash table: 2:delete hash table 3:search hash table 4:display hash table
5:exitenter your choice4 hash table elements are 0 0 78 67



LINEAR PROBING

1:insert hash table: 2:delete hash table 3:search hash table 4:display hash table
5:exitenter your choice5
```

WEEK- 11

Aim: Write C programs for implementing the following sorting methods:

a) Merge sort b) Heap sort.

// C Program to implement Merge sort

```
#include<stdio.h>

int a[100]; //globally declaring array

void merge (int a[],int low,int high,int mid);
void mergesort (int a[],int low,int high);
void main(){

int i,n;

printf("\n enter the no. of elements");
scanf("%d",&n);
printf("enter elements of list");
for(i=0;i<n;i++) //entering loop//
scanf("%d",&a[i]);
mergesort(a,0,n-1);//calling function//
printf("\n sorted array\n");
for(i=0;i<n;i++)//entering loop//
printf("%d\n",a[i]);
}
void mergesort(int a[100],int low,int high){

int mid;

if(low<high)//checing condition//{

mid=(low+high)/2;
```

```
mergesort(a,low,mid);

 mergesort(a,mid+1,high);

merge(a,low,high,mid);
}}

void merge(int a[100],int low,int high,int mid) //called function//

{

int i,j,k,c[50];
i=low;
j=mid+1;
k=low;
while((i<=mid)&&(j<=high))

{

if(a[i]<a[j]){

c[k]=a[i];
i++;
k++;
}

else{

c[k]=a[j];
j++;
k++;

}}

while(i<=mid){
c[k]=a[i];
i++;
k++;
```

```
}
while(j<=high){
c[k]=a[j];
j++;
k++;
}
for(i=0;i<k;i++)
 a[i]=c[i]; }
```

Output:

```
enter the no. of elements5 enter elements of list 23 45
12
23
31
sorted array 12 23 23
31
45
```

// C Program to implement Heap sort

```
#include<stdio.h>
void create(int []);
void hsort(int [],int);
int main()
{
int heap[30],n,i,last,temp;
printf("Enter no. of elements:");
scanf("%d",&n);
printf("\nEnter elements:");
for(i=1;i<=n;i++)
scanf("%d",&heap[i]);
//create a heap
heap[0]=n;
create(heap);
//sorting
while(heap[0] > 1)
{
//swap heap[1] and heap[last]
last=heap[0];
 temp=heap[1];
heap[1]=heap[last];
heap[last]=temp;
heap[0]--;
 hsort(heap,1);}
/print sorted data

printf("\nArray after sorting:\n");

for(i=1;i<=n;i++)
printf("%d ",heap[i]);
return 0;}
void create(int heap[])
{
int i,n;
n=heap[0]; //no. of elements
for(i=n/2;i>=1;i--)
hsort(heap,i);
}
void hsort(int heap[],int i) {

int j,temp,n,flag=1;

n=heap[0];
while(2*i<=n && flag==1) {
j=2*i; //j points to left child

if(j+1<=n && heap[j+1] > heap[j])
j=j+1;
 if(heap[i] > heap[j])
 flag=0;
else {

temp=heap[i];

heap[i]=heap[j];

heap[j]=temp;

i=j;  } }}
```

Output:

```
Enter no. of elements: 4
Enter elements: 10 9
56

32

Array after sorting:

9 10 32 56
```

WEEK-12

Aim: Write a C program to implement

i) Bubble sort ii) Selection sort

iii) quick sort iv) Insertion sort

//Program to implement Bubble sort

```
#include <stdio.h>

void bubblesort(int x[], int n)

{

int i, j, temp;

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

{

for (j = 0; j < n–i-1 ; j++)

{

if (x[j] > x[j+1])

{

temp=x[j];

x[j] = x[j+1];
x[j+1]= emp;
}}

}}
```

```
main(){
int i, n, x[25];
printf("\n Enter the number of elements: ");
scanf("%d",&n);
printf("\nEnter Data:");
for(i = 0; i < n ; i++)
scanf("%d",&x[i]);
bubblesort(x, n);
printf ("\n Array Elements after sorting: ");
for (i = 0; i < n; i++)
printf ("%5d", x[i]);
}
```

Output:

```
Enter number of elements 6
Enter elements into an array 23
12
2
34
90
56
Array Elements after sorting:
2    12    23    34    56    90
```

# Bubble sort in Python

```
def bubbleSort(array):

  # loop to access each array element
  for i in range(len(array)):

    # loop to compare array elements
    for j in range(0, len(array) - i - 1):

      # compare two adjacent elements
      # change > to < to sort in descending order
      if array[j] >array[j + 1]:

        # swapping elements if elements
        # are not in the intended order
        temp = array[j]
        array[j] = array[j+1]
        array[j+1] = temp

data = [-2, 45, 0, 11, -9]
bubbleSort(data)
print('Sorted Array in Ascending Order:')
print(data)
```

//Program to implement Selection sort

```
#include <stdio.h>

void ssort(int a[], int n)
{

Int i,j,k,min,temp;

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

{

if(a[min]>a[j])

min=j;
}

temp=a[i];

a[i]=a[min];

a[min]=temp;
}}

void main()

{

int i, n, x[25];

printf("\n Enter the number of elements: ");
scanf("%d",&n);
printf("\nEnter Data:");
for(i = 0; i < n ; i++)
scanf("%d",&x[i]);
ssort(x, n);
printf ("\n Array Elements after sorting: ");
for (i = 0; i < n; i++)
printf ("%5d", x[i]);

}
```

Output:

```
Enter the number of elements: 4 Enter Data:23

15

45

34

Array Elements after sorting: 15 23 34 45
```

# Selection sort in Python

```
def selectionSort(array, size):

   for step in range(size):
min_idx = step

      for i in range(step + 1, size):

         # to sort in descending order, change > to < in this line
         # select the minimum element in each loop
         if array[i] < array[min_idx]:
min_idx = i

      # put min at the correct position
      (array[step], array[min_idx]) = (array[min_idx], array[step])


data = [-2, 45, 0, 11, -9]
size = len(data)
selectionSort(data, size)
print('Sorted Array in Ascending Order:')
print(data)
```

//Program to implement Quick sort

```
#include<stdio.h>

void qsort(int a[10],int first,int last);

int main()
{
int i,n,a[10],j,pivot,last,t;

printf("enterthe no of elements\n");
scanf("%d",&n);
printf("enterthe elements\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
qsort(a,0,n-1);
printf("sortedelements is\n");
for(i=0;i<n;i++)
printf("\n%d",a[i]);
}

void qsort(int a[10],int first,int last)

{

int i,j,t,pivot,n;

if(first<last)

{i=first;

j=last;
pivot=first;
while(i<j)
{while(a[i]<=a[pivot]&&i<last)

i++;
while(a[j]>a[pivot])
 j--;
if(i<j)

{

t=a[i];
a[i]=a[j];
a[j]=t;
} }

t=a[pivot];
a[pivot]=a[j];

a[j]=t;

qsort(a,first,j-1);

qsort(a,j+1,last);
} }
```

Output:

```
enter the no of elements 6
enter the elements 23
12

14

2

78
67

sorted elements is
2 12
14
23

67

78
```

# Quick sort in Python

```
# function to find the partition position
def partition(array, low, high):

  # choose the rightmost element as pivot
  pivot = array[high]

  # pointer for greater element
i = low - 1

  # traverse through all elements
  # compare each element with pivot
  for j in range(low, high):
    if array[j] <= pivot:
      # if element smaller than pivot is found
      # swap it with the greater element pointed by i
i = i + 1

      # swapping element at i with element at j
      (array[i], array[j]) = (array[j], array[i])

  # swap the pivot element with the greater element specified by i
  (array[i + 1], array[high]) = (array[high], array[i + 1])

  # return the position from where partition is done
  return i + 1

# function to perform quicksort
def quickSort(array, low, high):
  if low < high:

    # find pivot element such that
    # element smaller than pivot are on the left
    # element greater than pivot are on the right
    pi = partition(array, low, high)

    # recursive call on the left of pivot
quickSort(array, low, pi - 1)

    # recursive call on the right of pivot
quickSort(array, pi + 1, high)


data = [8, 7, 2, 1, 0, 9, 6]
print("Unsorted Array")
print quickSort(data, 0, size - 1)
print('Sorted Array in Ascending Order:')
print(data)
```

```
 (data)

size = len(data)
```

//Program to implement Insertion sort

```
#include <stdio.h>

void isort(int a[], int n)
{

inti,j,temp;

for(i=1;i<n;i++)
{
temp=a[i];
j=i-1;
while((temp<a[j])&&(j>=0))

{a[j+1]=a[j];

j=j-1;
}
a[j+1]=temp; }}

void main(){

int i, n, x[25];

printf("\n Enter the number of elements: ");

scanf("%d",&n);

printf("\nEnter Data:");

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

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

isort(x, n);

printf ("\n Array Elements after sorting: ");

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

printf ("%5d", x[i]);

}
```

Output:

```
Enter the number of elements: 5

Enter Data:
11

2

56

45

34
Array Elements after sorting: 2 11 34 45 56
```

# Insertion sort in Python

```
def insertionSort(array):
    for step in range(1, len(array)): key
        = array[step]
        j = step - 1

        # Compare key with each element on the left of it until an element smaller than it is
found
        # For descending order, change key<array[j] to key>array[j]. while j >=
        0 and key < array[j]:
array[j + 1] = array[j] j =
            j - 1

        # Place key at after the element just smaller than it. array[j + 1]
= key

data = [9, 5, 1, 4, 3]
insertionSort(data)
print('Sorted Array in Ascending Order:')
print(data)
```

WEEK-13

Aim: Write C programs for implementing the following graph traversal algorithms: a)Depth first traversal

// program to implement DFS

```
#include <stdio.h> void dfs(int);
int g[10][10],visited[10],n,vertex[10];
void main(){
int i,j;
printf("enter number of vertices:");
scanf("%d",&n);
printf("enter the val of vertex:");
for(i=0;i<n;i++)
scanf("%d",&vertex[i]);
printf("\n enter adjecency matrix of the graph:");
 for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&g[i][j]);
for(i=0;i<n;i++)
visited[i]=0;
dfs(0); }

void dfs(int i) {

 int j;
printf("%d",vertex[i]);
visited[i]=1;
for(j=0;j<n;j++)
if(!visited[j]&&g[i][j]==1)
dfs(j);
}
```

<u>Output:</u>

```
enter number of vertices:4 enter the val of vertex: 1 2 3 4
enter adjecency matrix of the graph : 0 1 1 1

1 0 0 1


1 0 0 1


1 1 1 0


1 2 4 3
```

BEYOND SYLLABUS

Aim: Write a C program that uses stack operations to convert a given infix expression into its postfix Equivalent, Implement thestack using an array.

//Program to convert infix to postfix expression.

```
 #include<stdio.h>
 #include<string.h>
 #include<stdlib.h>
 #defineMAX 20
 char stack[MAX];
 int top=1;
 char pop(); /*declaration of pop function*/
 void push(char item); /*declaration of push function*/
 int prcd(char symsbol) /*checking the precedence*/
 {
 switch(symbol)     /*assigning values for symbols*/
 {
 case '+':
case '-': return 2;
 break;
 case '*':
case '/': return 4;
 break;
case    '^':return    6;
 break;
 case '(':
```

```
 case ')':

 case '#':return 1;
 break; }}

 int(isoperator(char symbol)) /*assigning operators*/

 {

 switch(symbol) {

 case '+':

 case '*':12

 case '-':

 case '/':

 case '^':

 case '(':

case ')': return 1; break;

default:return 0; }}

 /*converting infix to postfix*/

 void convertip(char infix[],char postfix[]) {

int i,symbol,j=0;

stack[++top]='#';

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

symbol=infix[i];
 if(isoperator(symbol)==0) {

 postfix[j]=symbol;

  j++;  }
```

```
else {
if(symbol=='(')
push(symbol); /*function call for pushing elements into the stack*/
else if(symbol==')') {
while(stack[top]!='(')  {
postfix[j]=pop();
j++; }
pop(); /*function call for popping elements into the stack*/
}
Else {
if(prcd(symbol)>prcd(stack[top]))
push(symbol);
 else {
while(prcd(symbol)<=prcd(stack[top])) {
postfix[j]=pop();
 j++; }
push(symbol);
}/*end of else loop*/
}/*end of else loop*/
} /*end of else loop*/
}/*end of for loop*/
While (stack[top]!='#') {
postfix[j]=pop();
j++;  }

postfix[j]='\0'; /*null terminate string*/

}

/*main program*/

void main() {

charinfix[20],postfix[20];

printf("enter the valid infix string \n");

gets(infix);
convertip(infix,postfix); /*function call for converting infix to postfix */
printf("the corresponding postfix string is:\n");
puts(postfix);  }

/*push operation*/

void push(char item) {

top++;

stack[top]=item;  }

/*pop operation*/

char pop() {

char a; a=stack[top];

top--;

return a; }
```

Output:

```
enter the valid infix string a+b*c

the corresponding postfix string is: abc*+
```

Aim : Write a C program to implement circular linked list.

// **program to implement circular linked list**

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *link;
}*cur,*prev,*temp,*last,*first=NULL;
void create();
void display();
void insert_begin();
void insert_after();
void insert_before();
void del_begin();
void del();
int main() {
Int ch=0;
 while(ch<=7){
printf("enter ur choice\n");
scanf("%d",&ch);
switch(ch) {
case1:create(); break;
case 2:insert_begin();

break;
case3:insert_after(); break;
case 4: insert_before(); break;
case5:del_begin(); break;
case 6:del(); break;
case 7:display(); break;
}//switch }//while }//main

void create() {

cur=(structnode*)malloc(sizeof(struct
node)); printf("enter data\n");
scanf("%d",&cur->data);
cur->link=NULL;
if(first==NULL){

first=cur; last=cur; }

else{

last->link=cur;
last=cur;}

last->link=first;

}//create

void insert_begin(){

cur=(struct node*)malloc(sizeof(struct node));

printf("enter data\n");
scanf("%d",&cur->data); last->link=cur;
cur->link=first; first=cur;
}//insert_begin()
```

```
void insert_after(){

int key;

cur=(struct node*)malloc(sizeof(struct node));

printf("enter data\n");
scanf("%d",&cur->data);
cur->link=NULL;
printf("enter   element   after   which u    want   to
insert\n"); scanf("%d",&key);
temp=first;
while(temp!=NULL) {

if(temp->data==key)

break;
else
temp=temp->link;  }

cur->link=temp->link; temp->link=cur;
}//insert_after
void insert_before() {

int key;

cur=(struct node*)malloc(sizeof(struct node));

printf("enter data\n");
scanf("%d",&cur->data);
cur->link=NULL;
printf("Enter before which u want to insert\n");
scanf("%d",&key);
temp=first;
while(temp!=NULL) {
if(temp->data==key) break;
else {
prev=temp;
temp=temp->link;
}//else
}//while
cur->link=temp;
prev->link=cur;}
void del_begin() {
temp=first; first=first->link;
last->link=first; free(temp); }
void del() {
int key;
temp=first;
printf("enter data to delete");
scanf("%d",&key);
while(temp!=NULL){
if(temp->data==key) break;
else{
prev=temp;
temp=temp->link;
}//else  }
prev->link=temp->link;
free(temp);     }//del
void display() {
```

```
temp=first;

do {

 printf("%d\t",temp->data);

 temp=temp->link;
 }while(temp!=first);

 }
```

Output:

```
 enter ur choice 1 23
enter ur  choice 2
32
 enter ur choice 3

 enter element after which u want to insert 32 43


 enter ur choice 4

 Enter before which u want to insert
 43 12
 enter ur choice 7 23->32->12->43
```

Aim: Write a C program to reverse the elements in the stack using recursion.

// Program to reverse the elements using stack

```
#include  <stdio.h>
#include <stdlib.h>
struct node
{
int a;
struct node *next;
};
void generate(struct node **);
void display(struct node *);
void stack_reverse(struct node **, struct node **);
 void delete(struct node **);
int main(){
struct node *head = NULL;
generate(&head);
printf("\nThesequence of contents in stack\n");
 display(head);
printf("\nInversing the contents of the stack\n");
 if (head != NULL){
stack_reverse(&head, &(head->next)); }
printf("\nThe contents in stack after reversal\n");
display(head);
delete(&head);
return 0;}
void stack_reverse(struct node **head, struct node **head_next) {
```

```
struct node *temp;

if (*head_next != NULL) {

temp = (*head_next)->next;

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

*head_next = temp;

stack_reverse(head, head_next); } }

void display(struct node *head) {

if (head != NULL) {

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

 display(head->next); }}

void generate(struct node **head)

{ int num, i;

struct node *temp;

printf("Enter length of list: ");

scanf("%d", &num);
for (i = num; i > 0; i--) {

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

temp->a = i;
if (*head == NULL){

*head = temp;

(*head)->next = NULL;}

else{
```

```
temp->next = *head;

*head = temp;  }}}

void delete(struct node **head)

{

struct node *temp;

while (*head != NULL)

{

temp = *head;

*head = (*head)->next;

free(temp);

}}
```

Output:

```
Enter length of list: 10

The sequence of contents in stack 1 2 3 4 5 6 7 8 9 10
Reversing the contents of the stack
The contents in stack after reversal 10 9 8 7 6 5 4 3 2 1
```

Aim: Write a C program to find the largest element in a given doubly linked list

// **program to find the largest element in a given dll**

```
#include <stdio.h>
 #include <stdlib.h>
 struct node
{
int num;
struct node *next;
struct node *prev;
};
void create(struct node **);
int max(struct node *);
void release(struct node **);
int main()
{
struct node *p = NULL;
int n;
printf("Enter data into the list\n");
create(&p);
n = max(p);
printf("The maximum number entered in the list is %d.\n", n);
release (&p);
return 0;}
int max(struct node *head)
{
struct node *max, *q;
```

```
q = max = head;
while (q != NULL){
if (q->num > max->num)
{
max = q; }
q = q->next;  }
return (max->num);
}
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;
temp->prev = NULL;
if (*head == NULL){
*head = temp; }
Else {
rear->next= temp;
temp->prev = rear; }
rear = temp;
printf("Do you wish to continue [1/0]: ");
scanf("%d", &ch);
} while (ch != 0);
printf("\n");
}
void release(struct node **head)
{
struct node *temp = *head;
*head = (*head)->next;
while ((*head) != NULL){
```

```
free(temp);
 temp = *head;
(*head) = (*head)->next;
}}
```

Output:

```
Enter data into the list Enter number: 12
Do you wish to continue [1/0]: 1
Enter number: 7
Do you wish to continue [1/0]: 1
 Enter number: 23
Do you wish to continue [1/0]: 1
Enter number: 4
Do you wish to continue [1/0]: 1
 Enter number: 1
Do you wish to continue [1/0]: 1
 Enter number: 16
Do you wish to continue [1/0]: 0

The maximum number entered in the list is 23
```