

**LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING
(AUTONOMOUS)**

Data Structures Lab Manual

I Year II Semester (R23)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision of the Institute:

To empower the students to become technologically vibrant, innovative and emotionally matured; and to train them to face the challenges of the quality-conscious globalized world economy.

Mission of the Institute:

1. To provide an environment most conducive to learning and to create a stimulating intellectual atmosphere on the campus.
2. To achieve Academic Excellence.
3. To ensure a holistic development of personality.
4. To spread education to rural areas.
5. To establish partnership between Institution & Industry.

Vision of the Department:

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department:

DM1: To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.

DM2: To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.

DM3: Expose the students to cutting edge technologies which enhance their employability and knowledge.

DM4: Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs):

PEO1: Pursue higher education, entrepreneurship and research to compete at global level. PEO2: Design and develop products innovatively in computer science and engineering and in other allied fields.

PEO3: Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all the levels with ethics and necessary attitude.

PEO4: Serve ever-changing needs of society with a pragmatic perception.

PROGRAM OUTCOMES(POs):

PO 1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	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.
PO 3	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.
PO 4	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.
PO 5	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.
PO 6	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.
PO 7	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.
PO 8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	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.
PO 11	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.
PO 12	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

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success of organization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO 3	To inculcate an ability to analyze, design and implement database applications.

1. Pre-requisites: Knowledge of C programming language

2. Course Educational Objectives (CEOs):

In this course student will learn about

Write implement algorithms various different data structures, implement stacks, queues and implement different tree structures.

3. Course Outcomes (COs): At the end of the course, the student will be able to:

CO1: Apply Linear Data Structures for organizing the data efficiently (**Apply-L3**)

CO2: Apply Non-Linear Data Structures to organize data efficiently (**Apply-L3**)

CO3: Develop and implement hashing techniques for solving problems. (**Apply-L3**)

CO4: Improve individual/team work skills, communication and report writing skills with ethical values.

4. Course Articulation Matrix:

Course Code	COs	Programme Outcomes												PSOs		
		1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
23CS52	CO1	3	2	2	1	3	-	-	-	-	-	-	-	3	3	3
	CO2	3	2	2	1	3	-	-	-	-	-	-	-	3	3	3
	CO3	3	2	2	1	3	-	-	-	-	-	-	-	3	3	3
	CO4	-	-	-	-	-	-	-	2	2	2	2	2	-	-	-
1 = Slight (Low) 2 = Moderate (Medium) 3-Substantial(High)																

Exercise1: Array Manipulation

- Write a program to reverse an array.
- C Program to implement the Searching Techniques – Linear & Binary Search
- C Program to implement Sorting Techniques – Bubble, Selection and Insertion sort

Exercise2: LinkedList Implementation

- Implement a singly linked list and perform insertion and deletion operations.
- Develop a program to reverse a linked list iteratively and recursively.
- Solve problems involving linked list traversal and manipulation.

Exercise3: Linked List Applications

- Create a program to detect and remove duplicates from a linked list.
- Implement a linked list to represent polynomials and perform addition.
- Implement a double-ended queue (deque) with essential operations.

Exercise4: Double Linked List Implementation

- Implement a doubly linked list and perform various operations to understand its properties and applications.
- Implement a circular linked list and perform insertion, deletion, and traversal.

Exercise5: Stack Operations

- i) Implement a stack using arrays and linked lists.
- ii) Write a program to evaluate a postfix expression using a stack.
- iii) Implement a program to check for balanced parentheses using a stack.

Exercise6: Queue Operations

- i) Implement a queue using arrays and linked lists.
- ii) Develop a program to simulate a simple printer queue system.
- iii) Solve problems involving circular queues.

Exercise7: Stack and Queue Applications

- i) Use a stack to evaluate an infix expression and convert it to postfix.
- ii) Create a program to determine whether a given string is a palindrome or not.
- iii) Implement a stack or queue to perform comparison and check for symmetry.

Exercise8: Binary Search Tree

- i) Implementing a BST using a Linked List.
- ii) Traversing of BST.

Exercise9: Hashing

- i) Implement a hash table with collision resolution techniques.
- ii) Write a program to implement a simple cache using hashing.

I.a)Write a C program to implement various operations on List using arrays.**Source Code:-**

```

#include<stdio.h>
#include<conio.h>
void create();
void display();
void insertion();
void deletion();
void updation();
void search();
void sort();
void merge();
int a[10],n,i,pos,value,j,k;
void main()
{
    int ch;
    //clrscr();
    printf("\n1.Create\n2.Display\n3.Insertion\n4.Deletion\n5.Updation\n6.
Search\n7.Sort\n8.Merge");
    while(1)
    {
        printf("\nEnter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:create();
                break;
            case 2:display();
                break;
            case 3:insertion();
                break;
            case 4:deletion();
                break;
            case 5:updation();
                break;
            case 6:search();
                break;
            case 7:sort();
                break;
            case 8:merge();
                break;
            default:exit(1);
        }
    }
}

```

```
        }
        getch();
    }

void create()
{
    printf("\nEnter size of the array\n");
    scanf("%d",&n);
    printf("\nEnter elements of the array\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}

void display()
{
    printf("\nElements of the array are\n");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
}

void insertion()
{
    printf("\nEnter the element to be inserted");
    scanf("%d",&value);
    printf("\nEnter the position at which the element is inserted\n");
    scanf("%d",&pos);
    n++;
    for(i=n-1;i>pos;i--)
        a[i]=a[i-1];
    a[pos]=value;
}

void deletion()
{
    printf("\nEnter the position at which the element is deleted\n");
    scanf("%d",&pos);
    n--;
    for(i=pos;i<n;i++)
        a[i]=a[i+1];
}

void updation()
{
    printf("\nEnter the position at which the element is updated\n");
    scanf("%d",&pos);
```

```
    printf("\nEnter the element to be updated\n");
    scanf("%d",&value);
    a[pos]=value;
}
```

```
void search()
{
    int key,flag=0;
    printf("\nEnter key element\n");
    scanf("%d",&key);
    for(i=0;i<n;i++)
    {
        if(a[i]==key)
        {
            flag=1;
            break;
        }
    }
    if(flag==1)
        printf("Key element is found");
    else
        printf("key element is not found");
}
```

```
void sort()
{
    int temp;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
}
```

```
void merge()
{
```

```

int b[5],c[10],n1;
printf("\nEnter size of 2nd array\n");
scanf("%d",&n1);
printf("\nEnter elements of 2nd array\n");
for(j=0;j<n1;j++)
scanf("%d",&b[j]);
for(i=0,k=0;i<n;i++,k++)
    c[k]=a[i];
for(j=0;j<n1;j++,k++)
    c[k]=b[j];
printf("\nAfter merging the elements of array are\n");
for(k=0;k<n+n1;k++)
printf("%d\t",c[k]);
}

```

Output:-

- 1.Create
- 2.Display
- 3.Insertion
- 4.Deletion
- 5.Updation
- 6.Search
- 7.Sort
- 8.Merge

Enter your choice

1

Enter size of the array

5

Enter elements of the array

2

3

5

8

9

Enter your choice

2

Elements of the array are

2 3 5 8 9

Enter your choice

3

enter the element to be inserted6

enter the position at which the element is inserted

5

Enter your choice

2

Elements of the array are

2 3 5 8 9 6

Enter your choice

4

Enter your choice

4

Enter the position at which the element is deleted

5

Enter your choice

2

Elements of the array are

2 3 5 8 9

Enter your choice

5

Enter the position at which the element is updated

4

Enter the element to be updated

99

Enter your choice

2

Elements of the array are

2 3 5 8 99

Enter your choice

6

Enter key element

99

Key element is found

Enter your choice

7

Enter your choice

2

Elements of the array are

2 3 5 8 99

Enter your choice

10

I.b) Write a C Program to implement Linear Search.**Source code:-**

```

#include<stdio.h>
#include<conio.h>
main( )
{
    int a[10],n,key,index,i,flag=0;
    clrscr( );
    printf("\nEnter size of the array");
    scanf("%d",&n);
    printf("\nEnter elements of the array");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nEnter key element");
    scanf("%d",&key);
    for(i=0;i<n;i++)
    {
        if(a[i]==key)
        {
            flag=1;
            break;
        }
    }
    if(flag==1)
        printf("\nKey element is found");
    else
        printf("\nKey element is not found");
    getch();
}

```

Output:-

```

Enter the size of the array 5
Enter the elements of the array
8 3 9 2 35
Enter key element
35

```

Key element is found.

I.b) Write a C Program to implement Binary Search.

Source code:-

```
#include<stdio.h>
#include<conio.h>
main( )
{
    int n,a[10],i,key,low,mid,high,flag;
    clrscr();
    printf("\nEnter size of the array");
    scanf("%d",&n);
    printf("\nEnter elements of the array");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nEnter key element");
    scanf("%d",&key);
    flag=0;
    low=0;
    high=n-1;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(key==a[mid])
        {
            flag=1;
            break;
        }
        else if(key<a[mid])
            high=mid-1;
        else
            low=mid+1;
    }
    if(flag==1)
        printf("key element is found");
    else
        printf("key element is not found");
}
```

```

    getch();
}

```

Output:-

```

Enter the size of the array 5
Enter the elements of the array
2 3 6 8 66
Enter key element
66
Key element is found.

```

I c)Write a C program to implement insertion sort, Bubble sort and Selection Sort..**Source code:-****Bubble sort:-**

```

#include<stdio.h>
#include<conio.h>
void bubble_sort(int [ ],int);
main()
{
    int n,a[10],i;
        clrscr();
        printf("\n Enter size of the array");
scanf("%d",&n);
        printf("\n Enter elements of the array");
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);
        bubble_sort(a,n);
        printf("\n After sorting the elements of the array are");
        for(i=0;i<n;i++)
            printf("%d \t",a[i]);
getch();
}

void bubble_sort(int a[],int n)
{
    int i,j,temp;
        for(i=0;i<n-1;i++)

```

```

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

```

Output:-

Enter size of the array5

Enter elements of the array

5 4 9 2 99

After sorting the

elements of the array are2 4 5 9 99

Insertion sort:-

```

#include<stdio.h>
#include<conio.h>
void insertion_sort(int [ ],int);
void main()
{
    int n,a[10],i;
    printf("\nEnter size of the array\nn");
    scanf("%d",&n);
    printf("Enter elements of the array");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    insertion_sort(a,n);
    printf("\nAfter sorting elements of the array are\n");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
    getch();
}

```

```
void insertion_sort(int a[],int n)
{
    int i,j,temp;
    for(i=1;i<n;i++)
    {
        temp=a[i];
        for(j=i;j>0 && a[j-1]>temp;j--)
            a[j]=a[j-1];
        a[j]=temp;
    }
}
```

Output:-

Enter size of the array6

Enter elements of the array

66 2 5 22 4 99

After sorting the elements of the array are2 4 5 22 66 99

2.a) Write a C program to implement various operations on Single linked List using pointers.

Source Code: -

```
#include<stdio.h>
#include<conio.h>
void create();
void display();
void insert_at_begin();
void insert_at_end();
void insert_at_pos();
void delete_at_begin();
void delete_at_end();
void delete_at_pos();
void reverse();
struct node
{
    int data;
    struct node *next;
}*new,*head,*temp,*tail;
int i,value,pos;
void main()
{
    int ch;
    char c;
    clrscr();
    printf("\n1.create\n2.Display\n3.Insert    at    begin\n4.Insert    at
end\n5.Insert at specified positon\n6.Delete at begin\n7.Delete at end\n8.Delete at
specified positon\n9.Reverse");
    while(1)
    {
        printf("\nEnter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:create();
                break;
            case 2:display();
                break;
            case 3:insert_at_begin();
                break;
            case 4:insert_at_end();
                break;
```

```

        case 5:insert_at_pos();
            break;
        case 6:delete_at_begin();
            break;
        case 7:delete_at_end();
            break;
        case 8:delete_at_pos();
            break;
        case 9:reverse();
            break;
        default:exit(0);
    }

}

}

void create()
{
    char c;
    do
    {
        new=(struct node *)malloc(sizeof(struct node));
        printf("\nEnter a value\n");
        scanf("%d",&value);
        new->data=value;
        new->next=NULL;
        if(head==NULL)
        {
            head=new;
            tail=new;
        }
        else
        {
            tail->next=new;
            tail=new;
        }
        printf("\nDo you want to add one more node to the list (y/n)?\n");
        fflush(stdin);
        c=getch();
    }while(c=='y');
}

```

```
}

void display()
{
    temp=head;
    printf("\nThe elements of list are\n");
    while(temp!=NULL)
    {
        printf("%d->",temp->data);
        temp=temp->next;
    }
}

void insert_at_begin()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\nEnter value\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
    new->next=head;
    head=new;
}

void insert_at_end()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\nEnter value\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
    tail->next=new;
    tail=new;
}

void insert_at_pos()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\nEnter the position\n");
    scanf("%d",&pos);
    printf("\nEnter value\n");
    scanf("%d",&value);
    new->data=value;
```

```
        new->next=NULL;
        temp=head;
        for(i=0;i<pos-1;i++)
        {
            temp=temp->next;
        }
        new->next=temp->next;
        temp->next=new;
    }

void delete_at_begin()
{
    temp=head;
    head=head->next;
    temp->next=NULL;
}

void delete_at_end()
{
    temp=head;
    while(temp->next!=tail)
    {
        temp=temp->next;
    }
    temp->next=NULL;
    tail=temp;
}

void delete_at_pos()
{
    printf("\nEnter the position\n");
    scanf("%d",&pos);
    temp=head;
    for(i=0;i<pos-1;i++)
    {
        temp=temp->next;
    }
    temp->next=temp->next->next;
}

void reverse()
{

```

```

    struct node *prev=NULL;
    struct node *current,*next;
    current=head;
    while(current!=NULL)
    {
        next=current->next;
        current->next=prev;
        prev=current;
        current=next;
    }
    head=prev;
}

```

Output:-

```

1.create
2.Display
3.Insert at begin
4.Insert at end
5.Insert at specified positon
6.Delete at begin
7.Delete at end
8.Delete at specified positon
9.Reverse
Enter your choice1
Enter the value 11
Do you want to add one more node to the list (y/n)? y
Enter the value 22
Do you want to add one more node to the list (y/n)? y

Enter the value 33
Do you want to add one more node to the list (y/n)? y

Enter the value 44
Do you want to add one more node to the list (y/n)? y

Enter the value 55
Do you want to add one more node to the list (y/n)? n
Enter your choice2
11->22->33->44->55->
Enter your choice3
Enter the value 66
Enter your choice2
66->11->22->33->44->55->
Enter your choice10

```

III.B) Write a C program to create a circular linked list so that the input order of data items is maintained. Add the following functions to carry out the following operations on circular single linked lists. a) Count the number of nodes. b) insert a node c) delete a node

Source Code:-

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct node
{
    int data;
    struct node *next;
}*head,*tail,*new,*temp;
int value,pos,i;
void create();
void display();
void insert_at_begin();
void insert_at_end();
void insert_at_pos();
void delete_at_begin();
void delete_at_end();
void delete_at_pos();
void count();

void main()
{
    int ch;
    clrscr();
    printf("\n1.create\n2.display\n3.insert_at_begin\n4.insert_at_end\n5.insert_at_pos\n6.delete_at_begin\n7.delete_at_end\n8.delete_at_pos\n9.count");
    while(1)
    {
        printf("\n enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: create();
                    break;
            case 2: display();
                    break;
```

```

        case 3: insert_at_begin();
            break;
        case 4: insert_at_end();
            break;
        case 5: insert_at_pos();
            break;
        case 6: delete_at_begin();
            break;
        case 7: delete_at_end();
            break;
        case 8: delete_at_pos();
            break;
        case 9: count( );
            break;
        default: exit(1);
    }

    }
    getch();
}

void create()
{
    char c;
    do
    {
        new=(struct node *)malloc(sizeof(struct node));
        printf("\n enter a value\n");
        scanf("%d",&value);
        new->data=value;
        new->next=NULL;
        if(head==NULL)
        {
            head=new;
            tail=new;
            tail->next=head;
        }
        else
        {
            tail->next=new;
            tail=new;
            tail->next=head;
        }
        printf("\n do you want to add one more node to the
list(y/n)?\n");
    }
    while(c=='y');
}

```

```
        fflush(stdin);
        scanf("%c",&c);
    }while(c=='y');
}
void display()
{
    printf("\n elements of linked list are\n");
    temp=head;
    while(temp->next!=head)
    {
        printf("%d->",temp->data);
        temp=temp->next;
    }
    printf("%d->",temp->data);
}
void insert_at_begin()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\n enter a value\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
    new->next=head;
    tail->next=new;
    head=new;
}
void insert_at_end()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\n enter a value\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
    tail->next=new;
    new->next=head;
    tail=new;
}
void insert_at_pos()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\n enter position\n");
    scanf("%d",&pos);
    printf("\n enter a value\n");
```

```

scanf("%d",&value);
new->data=value;
new->next=NULL;
temp=head;
for(i=0;i<pos-1;i++)
    temp=temp->next;
new->next=temp->next;
temp->next=new;
}
void delete_at_begin()
{
    temp=head;
    head=head->next;
    tail->next=head;
    temp->next=NULL;
}
void delete_at_end()
{
    temp=head;
    while(temp->next!=tail)
        temp=temp->next;
    tail->next=NULL;
    tail=temp;
    tail->next=head;
}
void delete_at_pos()
{
    printf("\n enter position\n");
    scanf("%d",&pos);
    temp=head;
    for(i=0;i<pos-1;i++)
        temp=temp->next;
    temp->next=temp->next->next;
}
void count()
{
    int c=0;
    temp=head;
    while(temp->next!=head)
    {
        c++;
        temp=temp->next;
    }
}

```

```
        c++;  
        printf("\nNumber of nodes in the list are %d",c);  
    }
```

Output:-

```
1.create  
2.display  
3.insert_at_begin  
4.insert_at_end  
5.insert_at_pos  
6.delete_at_begin  
7.delete_at_end  
8.delete_at_pos  
9.count  
  enter your choice  
1  
  enter a value  
11  
  enter your choice  
2  
elements of linked list are  
11->  
enter your choice  
3  
  enter a value  
22  
  enter your choice  
2  
elements of linked list are  
22->11->  
enter your choice  
4  
  enter a value  
55  
  enter your choice  
2  
elements of linked list are
```

22->11->55->

enter your choice

7

enter your choice

2

elements of linked list are

22->11->

enter your choice

9

Number of nodes in the list are 2

enter your choice 10

I.e) Write a C program to implement various operations on double linked list.**Source Code:-**

```

#include<stdio.h>
#include<conio.h>
#include<alloc.h>
struct node
{
    struct node *prev;
    int data;
    struct node *next;
}*head,*tail,*new,*temp;
int i,pos,value;
void create();
void display();
void insert_at_begin();
void insert_at_end();
void insert_at_pos();
void delete_at_begin();
void delete_at_end();
void delete_at_pos();
void reverse();
void main()
{
    int ch;
    clrscr();
    printf("\n1.create\n2.display\n3.insert_at_begin\n4.insert_at_end\n5.ins
ert_at_pos\n6.delete_at_begin\n7.delete_at_end\n8.delete_at_pos\n9.reverse\n");
    while(1)
    {
        printf("\n enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: create();
                    break;
            case 2: display();

```

```

        break;
    case 3: insert_at_begin();
        break;
    case 4: insert_at_end();
        break;
    case 5: insert_at_pos();
        break;
    case 6: delete_at_begin();
        break;
    case 7: delete_at_end();
        break;
    case 8: delete_at_pos();
        break;
    case 9: reverse();
        break;
    default: exit(1);
}

}

getch();
}

void create()
{
    char c;
    do
    {
        new=(struct node *)malloc(sizeof(struct node));
        printf("\n enter a value\n");
        scanf("%d",&value);
        new->prev=NULL;
        new->data=value;
        new->next=NULL;
        if(head==NULL)
        {
            head=new;
            tail=new;
        }
    }
}

```

```

        else
        {
            tail->next=new;
            new->prev=tail;
            tail=new;
        }
        printf("\n do you want to add one more node to the list
(y/n)?");

        fflush(stdin);
        scanf("%c",&c);

    }while(c=='y');
}

void display()
{
    printf("\n elements of linked list are\n");
    temp=head;
    while(temp!=NULL)
    {
        printf("%d<->",temp->data);
        temp=temp->next;
    }
}

void insert_at_begin()
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\n enter a value\n");
    scanf("%d",&value);
    new->prev=NULL;
    new->data=value;
    new->next=head;
    head->prev=new;
    head=new;
}

void insert_at_end()
{
    new=(struct node *)malloc(sizeof(struct node));

```

```
        printf("\n enter a value\n");
        scanf("%d",&value);
        tail->next=new;
        new->prev=tail;
        new->data=value;
        new->next=NULL;
        tail=new;
    }
    void insert_at_pos()
    {
        new=(struct node *)malloc(sizeof(struct node));
        printf("\n enter position\n");
        scanf("%d",&pos);
        printf("\n enter a value\n");
        scanf("%d",&value);
        temp=head;
        for(i=0;i<pos-1;i++)
            temp=temp->next;
        new->data=value;
        new->next=temp->next;
        temp->next->prev=new;
        temp->next=new;
        new->prev=temp;
    }
    void delete_at_begin()
    {
        temp=head;
        head=head->next;
        temp->next=NULL;
        head->prev=NULL;
    }
    void delete_at_end()
    {
        temp=tail;
        tail=tail->prev;
        temp->prev=NULL;
```

```

        tail->next=NULL;
    }
    void delete_at_pos()
    {
        printf("\n enter position\n");
        scanf("%d",&pos);
        temp=head;
        for(i=0;i<pos-1;i++)
            temp=temp->next;
        temp->next=temp->next->next;
        temp->next->prev=temp;
    }
    void reverse()
    {
        temp=tail;
        while(temp!=NULL)
        {
            printf("%d<->",temp->data);
            temp=temp->prev;
        }
    }
}

```

Output:-

```

1.create
2.display
3.insert_at_begin
4.insert_at_end
5.insert_at_pos
6.delete_at_begin
7.delete_at_end
8.delete_at_pos
9.reverse
enter your choice
1
enter a value
11
enter your choice
3
enter a value
22
enter your choice

```

```
2
elements of linked list are
22<->11<->
enter your choice
4
enter a value
66
enter your choice
2
elements of linked list are
2<->11<->66<->
enter your choice
7
enter your choice
2
elements of linked list are
22<->11<->
enter your choice
10
```

II.a&b)Write a C program to implement a stack using array &linked list in which Push, Pop and display can be performed.

Stack using array

Source Code:-

```
#include<stdio.h>
#include<conio.h>
#define SIZE 5
void push( );
void pop( );
void peep( );
void display( );
int stack[10], top= -1, ele;
main( )
{
    int ch;
    clrscr( );
    while(1)
    {
        printf("\nMENU");
        printf("\n1.push \n2.pop \n3.peep \n4.display \n5.exit");
        printf("\nEnter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: push( );
                    break;
            case 2: pop( );
                    break;
            case 3: peep( );
                    break;
            case 4: display( );
                    break;
            default:exit( );
        }
    }
    getch( );
}
```

```
}
```

```
void push( )
```

```
{
```

```
    if( top==SIZE-1)
```

```
        printf("\nStack is full");
```

```
    else
```

```
    {
```

```
        top++;
```

```
        printf("\nEnter the element to be inserted into the stack");
```

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

```
        stack[top]=ele;
```

```
    }
```

```
}
```

```
void pop( )
```

```
{
```

```
    if( top== -1)
```

```
        printf("\nStack is empty");
```

```
    else
```

```
    {
```

```
        ele=stack[top];
```

```
        printf("\nThe deleted element from the stack is %d",ele);
```

```
        top--;
```

```
    }
```

```
}
```

```
void peep( )
```

```
{
```

```
    if( top == -1)
```

```
        printf("\nStack is empty");
```

```

        else
        {
            ele=stack[top];
printf("\nThe top most element of the stack is %d",ele);
        }
    }

void display( )
{
    int i;
    if( top== -1)
        printf("\nStack is empty");
    else
    {
        printf("\nThe elements of the stack are:\n");
        for(i=top;i>=0;i--)
            printf("%d\n",stack[i]);
    }
}

```

Output:-

```

MENU
1.push
2.pop
3.peep
4.display
5.exit
Enter your choice1
Enter the element to be inserted into the stack11
MENU
1.push
2.pop
3.peep
4.display
5.exit
Enter your choice1
Enter the element to be inserted into the stack22
MENU
1.push
2.pop
3.peep
4.display
5.exit

```

Enter your choice4

The elements of the stack are:

22

11

MENU

1.push

2.pop

3.peep

4.display

5.exit

Enter your choice2

The deleted element from the stack is 22

MENU

1.push

2.pop

3.peep

4.display

5.exit

Enter your choice3

The top most element of the stack is 11

MENU

1.push

2.pop

3.peep

4.display

5.exit

Enter your choice4

The elements of the stack are:

11

Stack using Linked Lists

Source Code:-

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
#include<alloc.h>
```

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node *next;
```

```
}*tos,*new,*temp;
```

```
void push( );
```

```
void pop( );
```

```
void peep( );
```

```
void display( );
```

```
int ele;
```

```
main( )
```

```
{
```

```

int ch;
clrscr( );
while(1)
{
    printf("\nMENU");
    printf("\n1.push \n2.pop \n3.peep \n4.display \n5.exit");
    printf("\nEnter your choice");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: push( );
                break;
        case 2: pop( );
                break;
        case 3: peep( );
                break;
        case 4: display( );
                break;
        default:exit( );
    }
}

getch( );
}

void push( )
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\nEnter an element to be inserted into the stack");
    scanf("%d",&ele);
    new->data=ele;
    new->next=NULL;
    if(tos==NULL)
        tos=new;
    else
    {
        new->next=tos;
    }
}

```

```
        tos=new;

    }

}

void pop( )
{
    if( tos==NULL)
        printf("\nStack is empty");

    else
    {
        temp=tos;
        ele=tos->data;
        printf("\nThe deleted element from the stack is %d",ele);
        tos=tos->next;
        temp->next=NULL;
    }
}

void peep( )
{
    if( tos==NULL)
        printf("\nStack is empty");

    else
    {
        ele=tos->data;
        printf("\nThe top most element of the stack is %d",ele);
    }
}

void display( )
{
    if( tos==NULL)
        printf("\nStack is empty");

    else
    {
```

```

printf("\nThe elements of the stack are:\n");
temp=tos;
while(temp!=NULL)
{
    printf("%d\n",temp->data);
    temp=temp->next;
}
}
}

```

Output:-

```

MENU
1.push
2.pop
3.peep
4.display
5.exit
Enter your choice1
Enter the element to be inserted into the stack11
MENU
1.push
2.pop
3.peep
4.display
5.exit
Enter your choice1
Enter the element to be inserted into the stack22
MENU
1.push
2.pop
3.peep
4.display
5.exit
Enter your choice4
The elements of the stack are:
22
11
MENU
1.push
2.pop
3.peep
4.display
5.exit
Enter your choice2
The deleted element from the stack is 22
MENU
1.push
2.pop
3.peep
4.display

```

5.exit

Enter your choice3

The top most element of the stack is 11

MENU

1.push

2.pop

3.peep

4.display

5.exit

Enter your choice4

The elements of the stack are:

11

II.b) Write a program to convert infix expression to post fix expressions using array implementation of stack

Source Code:-

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void push(char);
char pop();
int priority(char);
int top=-1;
char stack[20];
void main()
{
    char postfix[30],infix[30],ch;
    int i,j=0;
    clrscr();
    printf("\nEnter infix expression\n");
    gets(infix);
    for(i=0;infix[i]!='\0';i++)
    {
        ch=infix[i];
        if(ch=='(')
            push(ch);
        else if(ch==')')
        {
            while( stack[top]!='(' && top!=-1)
            {
                postfix[j]=pop();
                j++;
            }
            pop();
        }
        else if(isalpha(ch) || isdigit(ch))
        {
            postfix[j]=ch;
            j++;
        }
    }
}
```

```

        }
    else
    {
        while(priority(ch)<=priority(stack[top]) && top!=-1)
        {
            postfix[j]=pop();
            j++;
        }
        push(ch);
    }
}

while(top!=-1)
{
    postfix[j]=pop();
    j++;
}

postfix[j]='\0';
printf("\nresult is %s",postfix);
getch();
}

void push(char x)
{
    top++;
    stack[top]=x;
}

char pop()
{
    return stack[top--];
}

int priority(char ch)
{
    if(ch=='(')

```

```
        return 0;
    else if(ch=='+' || ch=='-')
        return 1;
    else if(ch=='*' || ch=='/' || ch=='%')
        return 2;
    else if(ch=='^')
        return 3;
}
```

Output:

Enter infix expression

a+b*c-d/f

result is abc*+df/-

II.c)Write a program for evaluating postfix expressions using array implementation of stack**Source code:-**

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
int stack[20];
char postfix[20],ch;
void push(int);
int pop();
int top=-1;

main()
{
    int i,op1,op2;
    clrscr();
    printf("\nEnter postfix expression\n");
    gets(postfix);
    for(i=0;postfix[i]!='\0';i++)
    {
        ch=postfix[i];
        if(isdigit(ch))
            push(ch-48);
        else
        {
            op2=pop();
            op1=pop();
            switch(ch)
            {
                case '+':push(op1+op2);break;
                case '-':push(op1-op2);break;
                case '*':push(op1*op2);break;
                case '/':push(op1/op2);break;
                case '%':push(op1%op2);break;
            }
        }
    }
}
```

```
    }  
    printf("\nThe result of postfix expression is %d",pop());  
    getch();  
}  
void push(int x)  
{  
    top++;  
    stack[top]=x;  
}  
int pop()  
{  
    return stack[top--];  
}
```

Output:-

enter postfix expression

234+*

The result of postfix expression is 14

II.d)Write a C program to implement a queue using arrays and linked list in which insertions, deletions and display can be performed.

Source code: -

Queue using Arrays:

```
#include<stdio.h>
#include<conio.h>
#define SIZE 5
void enqueue( );
void dequeue( );
void display( );
int queue[SIZE],front=-1,rear=-1,ele;
main( )
{
    int ch;
    while(1)
    {
        printf("\nMENU\n");
        printf("\n1.enqueue\n2.dequeue\n3.display");
        printf("\nenter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:enqueue( );
                    break;
            case 2:dequeue( );
                    break;
            case 3:display( );
                    break;
            default:exit( );
        }
    }
    getch();
}

void enqueue( )
{
```

```
        if(rear==SIZE-1)
            printf("\nQueue is full");
        else
        {
            rear++;
            printf("\nEnter an element to be inserted into the queue");
            scanf("%d",&ele);
            queue[rear]=ele;
            if(front==-1)
                front=0;
        }
    }

void dequeue( )
{
    if(front==-1 )
        printf("\nQueue is empty");
    else
    {
        ele=queue[front];
        printf("\nDeleted element from the queue is %d",ele);
        front++;
    }
}

void display( )
{
    int i;
    if(front==-1 )
        printf("\nQueue is empty");
    else
    {
        printf("\nThe elements of queue are\n");
        for(i=front;i<=rear;i++)
            printf("%d\t",queue[i]);
    }
}
```

```
}
```

Output:

MENU

1.enqueue

2.dequeue

3.display

enter your choice1

enter an element to be inserted into the queue11

MENU

1.enqueue

2.dequeue

3.display

enter your choice1

enter an element to be inserted into the queue22

MENU

1.enqueue

2.dequeue

3.display

enter your choice3

The elements of queue are11 22

MENU

1.enqueue

2.dequeue

3.display

enter your choice2

Deleted element from the queue is 11

MENU

1.enqueue

2.dequeue

3.display

enter your choice3

The elements of queue are22

MENU1.enqueue

2.dequeue

3.display

enter your choice5

Source code: -**Queue using Linked Lists:**

```
#include<stdio.h>
#include<conio.h>
void enqueue( );
void dequeue( );
void display( );
int value;
struct node
{
    int data;
    struct node *next;
}*front,*rear,*new,*temp;
main( )
{
    int ch;
    while(1)
    {
        printf("\nMENU\n");
        printf("\n1.enqueue\n2.dequeue\n3.display");
        printf("\nenter your choice");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:enqueue( );
                    break;
            case 2:dequeue( );
                    break;
            case 3:display( );
                    break;
            default:exit( );
        }
    }
    getch();
}
```

```
void enqueue( )
{
    new=(struct node *)malloc(sizeof(struct node));
    printf("\nEnter an element to be inserted into queue\n");
    scanf("%d",&value);
    new->data=value;
    new->next=NULL;
    if(rear==NULL)
        front=rear=new;
    else
    {
        rear->next=new;
        rear=new;
    }
}

void dequeue( )
{
    if(front==NULL)
        printf("\nQueue is empty");
    else
    {
        temp=front;
        printf("\nDeleted element from the queue is %d",front->data);
        front=front->next;
        temp->next=NULL;
    }
}

void display( )
{
    int i;
    if(front==NULL )
        printf("\nQueue is empty");
    else
    {
```

```

        printf("\nThe elements of queue are\n");
        temp=front;
        while(temp!=NULL)
        {
            printf("%d\t",temp->data);
            temp=temp->next;
        }
    }
}

```

Output:-

MENU

1.enqueue

2.dequeue

3.display

enter your choice1

enter an element to be inserted into the queue11

MENU

1.enqueue

2.dequeue

3.display

enter your choice1

enter an element to be inserted into the queue22

MENU

1.enqueue

2.dequeue

3.display

enter your choice3

The elements of queue are11 22

MENU

1.enqueue

2.dequeue

3.display

enter your choice2

Deleted element from the queue is 11

MENU

1.enqueue

2.dequeue

3.display

enter your choice3

The elements of queue are22

MENU1.enqueue

2.dequeue

3.display

enter your choice5

V.a)Write a C program to construct a binary tree and do inorder, preorder and post order traversals, printing the sequence of nodes visited in each case.

Source code: -

```
#include <stdio.h>
#include <stdlib.h>
struct tnode
{
    int data;
    struct tnode *left, *right;
};
struct tnode *root = NULL;
/* creating node of the tree and fill the given data */
struct tnode * createNode(int data) {
    struct tnode *newNode;
    newNode = (struct tnode *) malloc(sizeof(struct tnode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return (newNode);
}

/* inserting a new node into the tree */
void insertion(struct tnode **node, int data) {
    if (!*node) {
        *node = createNode(data);
    } else if (data < (*node)->data) {
        insertion(&(*node)->left, data);
    } else if (data > (*node)->data) {
        insertion(&(*node)->right, data);
    }
}

/* post order tree traversal */
void postOrder(struct tnode *node) {
    if (node) {
        postOrder(node->left);
```

```
postOrder(node->right);
printf("%d ", node->data);
    }
    return;
}

/* pre order tree traversal */
void preOrder(struct tnode *node) {
    if (node) {
printf("%d ", node->data);
preOrder(node->left);
preOrder(node->right);
    }
    return;
}

/* inorder tree traversal */
void inOrder(struct tnode *node) {
    if (node) {
inOrder(node->left);
printf("%d ", node->data);
inOrder(node->right);
    }
    return;
}

int main() {
    int data, ch;
    while (1) {
printf("\n1. Insertion\n2. Pre-order\n");
printf("3. Post-order\n4. In-order\n");
printf("5. Exit\nEnter your choice:");
scanf("%d", &ch);
        switch (ch) {
            case 1:
printf("Enter ur data:");
```

```

scanf("%d", &data);
insertion(&root, data);
                break;
            case 2:printf("Preorder is:");
preOrder(root);
                break;
            case 3:printf("Postorder is:");
postOrder(root);
                break;
            case 4:printf("Inorder is:");
inOrder(root);
                break;
            case 5:
exit(0);
            default:
printf("U've entered wrong opetion\n");
                break;
        }
    }
    return 0;
}

```

Output:

1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit

Enter your choice: 1

Enter ur data: 33

Enter your choice: 1

Enter ur data: 22

Enter your choice: 1

Enter ur data: 44

Enter your choice: 1

Enter ur data: 2

Enter your choice: 1

Enter ur data:99

Enter your choice:1

Enter ur data:3

Enter your choice:1

Enter ur data:88

Enter your choice:2

Preorder is :33 222 344 99 88

Enter your choice:3

Postorderis :3 222 8899 44 33

Enter your choice:4

Inorderis :2 322 3344 88 99

Enter your choice:5

V.b)Write a C program to implement BST operations- insertion, search and deletion.**Source code:**

```
#include <stdio.h>

#include <stdlib.h>

struct treeNode {
    int data;
    struct treeNode *left, *right;
};

struct treeNode *root = NULL;

/* create a new node with the given data */
struct treeNode* createNode(int data) {
    struct treeNode *newNode;
    newNode = (struct treeNode *) malloc(sizeof (struct treeNode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return(newNode);
}

/* insertion in binary search tree */
void insertion(struct treeNode **node, int data) {
    if (*node == NULL) {
        *node = createNode(data);
    } else if (data < (*node)->data) {
        insertion(&(*node)->left, data);
    } else if (data > (*node)->data) {
        insertion(&(*node)->right, data);
    }
}

/* deletion in binary search tree */
```

```

void deletion(struct treeNode **node, struct treeNode **parent, int data) {
    struct treeNode *tmpNode, *tmpParent;
    if (*node == NULL)
        return;
    if ((*node)->data == data) {
        /* deleting the leaf node */
        if (!(*node)->left && !(*node)->right) {
            if (parent) {
                /* delete leaf node */
                if ((*parent)->left == *node)
                    (*parent)->left = NULL;
                else
                    (*parent)->right = NULL;
                free(*node);
            } else {
                /* delete root node with no children */
                free(*node);
            }
        }
        /* deleting node with one child */
        } else if (!(*node)->right && (*node)->left) {
            /* deleting node with left child alone */
            tmpNode = *node;
            (*parent)->right = (*node)->left;
            free(tmpNode);
            *node = (*parent)->right;
        } else if ((*node)->right && !(*node)->left) {
            /* deleting node with right child alone */
            tmpNode = *node;
            (*parent)->left = (*node)->right;
            free(tmpNode);
            (*node) = (*parent)->left;
        } else if (!(*node)->right->left) {
            /*
             * deleting a node whose right child
             * is the smallest node in the right
             * subtree for the node to be deleted.

```

```

        */

tmpNode = *node;

(*node)->right->left = (*node)->left;

(*parent)->left = (*node)->right;
free(tmpNode);
*node = (*parent)->left;
} else {
    /*
     * Deleting a node with two children.
     * First, find the smallest node in
     * the right subtree. Replace the
     * smallest node with the node to be
     * deleted. Then, do proper connections
     * for the children of replaced node.
     */
    tmpNode = (*node)->right;
    while (tmpNode->left) {
        tmpParent = tmpNode;
        tmpNode = tmpNode->left;
    }
    tmpParent->left = tmpNode->right;
    tmpNode->left = (*node)->left;
    tmpNode->right = (*node)->right;
    free(*node);
    *node = tmpNode;
}
} else if (data < (*node)->data) {
    /* traverse towards left subtree */
    deletion(&(*node)->left, node, data);
} else if (data > (*node)->data) {
    /* traversing towards right subtree */
    deletion(&(*node)->right, node, data);
}

```

```

}

/* search the given element in binary search tree */
void findElement(struct treeNode *node, int data) {
    if (!node)
        return;
    else if (data < node->data) {
findElement(node->left, data);
    } else if (data > node->data) {
findElement(node->right, data);
    } else
printf("data found: %d\n", node->data);
    return;

}

void traverse(struct treeNode *node) {
    if (node != NULL) {
        traverse(node->left);
printf("%3d", node->data);
        traverse(node->right);
    }
    return;
}

int main() {
    int data, ch;
    while (1) {
printf("1. Insertion in Binary Search Tree\n");
printf("2. Deletion in Binary Search Tree\n");
printf("3. Search Element in Binary Search Tree\n");
printf("4. Inorder traversal\n5. Exit\n");
printf("Enter your choice:");
scanf("%d", &ch);
        switch (ch) {
            case 1:

```

```
        while (1) {  
            printf("Enter your data:");  
            scanf("%d", &data);  
            insertion(&root, data);  
            printf("Continue Insertion(0/1):");  
            scanf("%d", &ch);  
            if (!ch)  
                break;  
        }  
        break;  
    case 2:  
        printf("Enter your data:");  
        scanf("%d", &data);  
        deletion(&root, NULL, data);  
        break;  
    case 3:  
        printf("Enter value for data:");  
        scanf("%d", &data);  
        findElement(root, data);  
        break;  
    case 4:  
        printf("Inorder Traversal:\n");  
        traverse(root);  
        printf("\n");  
        break;  
    case 5:  
        exit(0);  
    default:  
        printf("u've entered wrong option\n");  
        break;  
    }  
}  
return 0;  
}
```

Output:-

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:1

Enter your data:11

Continue Insertion(0/1):1

Enter your data:22

Continue Insertion(0/1):1

Enter your data:15

Continue Insertion(0/1):1

Enter your data:76

Continue Insertion(0/1):1

Enter your data:42

Continue Insertion(0/1):1

Enter your data:75

Continue Insertion(0/1):1

Enter your data:94

Continue Insertion(0/1):1

Enter your data:36

Continue Insertion(0/1):0

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:4

Inorder Traversal:

11 15 22 36 42 75 76 94

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:3

Enter value for data:75

data found: 75

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:3

Enter value for data:75

data found: 75

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:3

Enter value for data:100

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:2

Enter your data:22

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree
3. Search Element in Binary Search Tree
4. Inorder traversal
5. Exit

Enter your choice:

4

Inorder Traversal:

11 15 36 42 75 76 94

1. Insertion in Binary Search Tree
2. Deletion in Binary Search Tree

3. Search Element in Binary Search Tree

4. Inorder traversal

5. Exit

Enter your choice:5

VI.a)Write a C program to implement the following graph Traversals a) DFS b)**BFS****Source code:****DFS:**

```
#include<stdio.h>

void DFS(int);

int G[10][10],visited[10],n;    //n is no of vertices and graph is sorted in array
G[10][10]

void main()
{
    int i,j;
    printf("Enter number of vertices:");

    scanf("%d",&n);

    //read the adjacency matrix
    printf("\nEnteradjecency matrix of the graph:");

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&G[i][j]);

    //visited is initialized to zero
    for(i=0;i<n;i++)
        visited[i]=0;

    DFS(0);
}

void DFS(int i)
{
    int j;
    printf("\n%d",i);
    visited[i]=1;
```

```

    for(j=0;j<n;j++)
        if(!visited[j]&&G[i][j]==1)
            DFS(j);
}

```

Output:-

Enter number of vertices:3

Enter adjacency matrix of the graph:

0 1 1

1 0 1

1 1 0

0

1

2

BFS:-**Soucecode:-**

```
#include<stdio.h>
```

```
int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
```

```

void bfs(int v) {
for(i = 1; i<= n; i++)
    if(a[v][i] && !visited[i])
        q[++r] = i;
if(f <= r) {
    visited[q[f]] = 1;
    bfs(q[f++]);
}
}

```

```

void main() {
    int v;
    printf("\n Enter the number of vertices:");
    scanf("%d", &n);

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

```

```

q[i] = 0;
visited[i] = 0;
}

printf("\n Enter graph data in matrix form:\n");
for(i=1; i<=n; i++) {
    for(j=1;j<=n;j++) {
        scanf("%d", &a[i][j]);
    }
}

printf("\n Enter the starting vertex:");
scanf("%d", &v);
bfs(v);
printf("\n The node which are reachable are:\n");

for(i=1; i<= n; i++) {
    if(visited[i])
        printf("%d\t", i);
    else {
        printf("\n Bfs is not possible. Not all nodes are reachable");
        break;
    }
}

```

Output:-

Enter the number of vertices:4

Enter graph data in matrix form:

0 1 1 1

1 0 1 1

1 1 0 1

1 1 1 0

Enter the starting vertex:2

The node which are reachable are:

1 2 3 4

