INDEX

Ex. No.	Date	Title of the experiment	Page No	Total (25)	Sign

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
int linearsearch(int a[],int n,int key)
{
 int i;
 for(i=0;i<n;i++)
 {
 if(a[i]==key)
 {
 return i;
 }
 }
 return -1;
}
int main()
{
 int a[100],n,key,pos,i;
 printf("enter n value");
 scanf("%d",&n);
 printf("enter %d elements into array",n);
 for(i=0;i<n;i++)
 {
 scanf("%d",&a[i]);
 }
 printf("enter search key elements");
 scanf("%d",&key);
 pos=linearsearch(a,n,key);
 if(pos>=0)
 printf("element found at %d position",pos+1);
 else
 printf("element not found");
 return 1;
}
```

OUTPUT

enter n value:6

enter 6 elements into array:14 24 34 74 94 164

enter search key elements:74

element found at 4 position

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
int linear_search_with_recursion(int arr[], int value, int index, int n)
{
 int position = 0;
 if(index >= n)
 {
 return 0;
 }
 else if (arr[index] == value)
 {
 position = index + 1;
 return position;
 }
 else
 {
 return linear_search_with_recursion(arr, value, index+1, n);
 }
 return position;
}
int main()
{
 int n, value, position, m = 0, arr[100];
 printf("Enter the total elements in the array that you would like to take:");
 scanf("%d", &n);
 printf("Enter the elements of the array:\n");
 for (int i = 0; i < n; i++)
 {
 scanf("%d", &arr[i]);
 }
 printf("Enter the element you would like to search in the array: ");
 scanf("%d", &value);
 position = linear_search_with_recursion(arr, value, 0, n);
 if (position != 0)
 {
 printf("Element found at position %d ", position);
 }
 else
 {
 printf("Element not found in the array");
 }
 return 0;
}
```

OUTPUT

Enter the total elements in the array that you would like to take:5

Enter the elements of the array:

10 20 30 40 50

Enter the element you would like to search in the array: 40

Element found at position 4

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[20], i, n, key, low, high, mid;
 printf("Enter number of elements:");
 scanf("%d",&n);
 printf("Enter the array elements in ascending order:");
 for(i = 0; i < n; i++)
 {
 scanf("%d", &a[i]);
 }
 printf("Enter the key element:\n");
 scanf("%d", &key);
 low = 0;
 high = n - 1;
 while(high >= low)
 {
 mid = (low + high) / 2;
 if(key == a[mid])
 break;
 else
 {
 if(key > a[mid])
 low = mid + 1;
 else
 high = mid - 1;
 }
 }
 if(key == a[mid])
 printf("The key element is found at location %d", mid + 1);
 else
 printf("The key element is not found");
}
```

OUTPUT

Enter number of elements:5
Enter the array elements in ascending order:11 12 13 14 15
Enter the key element:
13
The key element is found at location 3

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#define size 10
int binsearch(int[], int, int, int);
int main() {
 int num, i, key, position;
 int low, high, list[size];
 printf("Enter the total number of elements:");
 scanf("%d", &num);
 printf("Enter the elements of list :");
 for (i = 0; i < num; i++) {
 scanf("%d", &list[i]);
 }
 low = 0;
 high = num - 1;
 printf("Enter element to be searched : ");
 scanf("%d", &key);
 position = binsearch(list, key, low, high);
 if (position != -1) {
 printf("Number present at %d", (position + 1));
 } else
 printf("The number is not present in the list");
 return (0);
}
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);
 }
}
```

OUTPUT

```
Enter the total number of elements:4
Enter the elements of list :15 35 55 75
Enter element to be searched : 35
Number present at 2
```

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
int main(){
 int arr[50], num, x, y, temp;
 printf("Please Enter the Number of Elements you want in the array: ");
 scanf("%d", &num);
 printf("Please Enter the Value of Elements: ");
 for(x = 0; x < num; x++)
 scanf("%d", &arr[x]);
 for(x = 0; x < num - 1; x++){
 for(y = 0; y < num - x - 1; y++){
 if(arr[y] > arr[y + 1]){
 temp = arr[y];
 arr[y] = arr[y + 1];
 arr[y + 1] = temp;
 }
 }
 }
 printf("Array after implementing bubble sort: ");
 for(x = 0; x < num; x++){
 printf("%d ", arr[x]);
 }
 return 0;
}
```

OUTPUT

Please Enter the Number of Elements you want in the array: 5

Please Enter the Value of Elements: 94 169 74 12 56

Array after implementing bubble sort: 12 56 74 94 169

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
int main()
{
 int a[100], n, i, j, position, swap;
 printf("Enter number of elements:");
 scanf("%d", &n);
 printf("Enter %d Numbers:", n);
 for (i = 0; i < n; i++)
 scanf("%d", &a[i]);
 for(i = 0; i < n - 1; i++)
 {
 position=i;
 for(j = i + 1; j < n; j++)
 {
 if(a[position] > a[j])
 position=j;
 }
 if(position != i)
 {
 swap=a[i];
 a[i]=a[position];
 a[position]=swap;
 }
 }
 printf("Sorted Array:");
 for(i = 0; i < n; i++)
 printf("%d\t", a[i]);
 return 0;
}
```

OUTPUT

Enter number of elements: 5
Enter 5 Numbers:25 45 15 05 95
Sorted Array: 5 15 25 45 95

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
int main(void)
{
 int n, i, j, temp;
 int arr[64];
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
 for (i = 0; i < n; i++)
 {
 scanf("%d", &arr[i]);
 }
 for (i = 1; i < n; i++)
 {
 j = i;
 while (j > 0 && arr[j - 1] > arr[j])
 {
 temp = arr[j];
 arr[j] = arr[j - 1];
 arr[j - 1] = temp;
 j--;
 }
 }
 printf("Sorted list in ascending order:\n");
 for (i = 0; i < n; i++)
 {
 printf("%d\n", arr[i]);
 }
 return 0;
}
```

OUTPUT

Enter number of elements:5

Enter 5 integers:

12 32 98 65 61 78

Sorted list in ascending order:

12

32

61

65

98

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
void ShellSort(int a[], int n)
{
 int i, j, increment, tmp;
 for(increment = n/2; increment > 0; increment /= 2)
 {
 for(i = increment; i < n; i++)
 {
 tmp = a[i];
 for(j = i; j >= increment; j -= increment)
 {
 if(tmp < a[j-increment])
 a[j] = a[j-increment];
 else
 break;
 }
 a[j] = tmp;
 }
 }
}
int main()
{
 int i, n, a[10];
 printf("Enter the number of elements : ");
 scanf("%d",&n);
 printf("Enter the elements : ");
 for(i = 0; i < n; i++)
 {
 scanf("%d",&a[i]);
 }
 ShellSort(a,n);
 printf("The sorted elements are : ");
 for(i = 0; i < n; i++)
 printf("%d ",a[i]);
 printf("\n");
 return 0;
}
```

OUTPUT

Enter the number of elements : 6

Enter the elements : 33 44 22 55 11 66

The sorted elements are : 11 22 33 44 55 66

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
 int main()
 {
 int arr[10], no, i, j, c, heap_root, temp;
 printf("Input number of elements: ");
 scanf("%d", &no);
 printf("Input array values one by one : ");
 for (i = 0; i < no; i++)
 scanf("%d", &arr[i]);
 for (i = 1; i < no; i++)
 {
 c = i;
 do
 {
 heap_root = (c - 1) / 2;
 /* to create MAX arr array */
 if (arr[heap_root] < arr[c])
 {
 temp = arr[heap_root];
 arr[heap_root] = arr[c];
 arr[c] = temp;
 }
 c = heap_root;
 } while (c != 0);
 }
 printf("Heap array : ");
 for (i = 0; i < no; i++)
 printf("%d\t ", arr[i]);
 for (j = no - 1; j >= 0; j--)
 {
 temp = arr[0];
 arr[0] = arr[j];
 arr[j] = temp;
 heap_root = 0;
 do
 {
 c = 2 * heap_root + 1;
 if ((arr[c] < arr[c + 1]) && c < j-1)
 c++;
 if (arr[heap_root]<arr[c] && c<j)
 {
 temp = arr[heap_root];
 arr[heap_root] = arr[c];
 arr[c] = temp;
 }
 heap_root = c;
 } while (c < j);
```

```
 }
 printf("\nSorted array : ");
 for (i = 0; i < no; i++)
 printf("\t%d", arr[i]);
 printf("\n");
}
```

OUTPUT

Input number of elements: 4

Input array values one by one : 200 400 300 100

Heap array : 400 200 300 100

Sorted array : 100 200 300 400

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define SIZE 10
void push(int);
void pop();
void display();
int stack[SIZE], top = -1;
void main()
{
 int value, choice;
 clrscr();
 while(1){
 printf("\n\n***** MENU *****\n");
 printf("1. Push\n2. Pop\n3. Display\n4. Exit");
 printf("\nEnter your choice: ");
 scanf("%d",&choice);
 switch(choice){
 case 1: printf("Enter the value to be insert: ");
 scanf("%d",&value);
 push(value);
 break;
 case 2: pop();
 break;
 case 3: display();
 break;
 case 4: exit(0);
 default: printf("\nWrong selection!!! Try again!!!");
 }
 }
}
void push(int value){
 if(top == SIZE-1)
 printf("\nStack is Full!!! Insertion is not possible!!!");
 else{
 top++;
 stack[top] = value;
 printf("\nInsertion success!!!");
 }
}
void pop(){
 if(top == -1)
 printf("\nStack is Empty!!! Deletion is not possible!!!");
 else{
 printf("\nDeleted : %d", stack[top]);
 top--;
 }
```

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

OUTPUT

***** MENU *****

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to be insert: 100

Insertion success!!!

***** MENU *****

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to be insert: 200

Insertion success!!!

***** MENU *****

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to be insert: 300

Insertion success!!!

***** MENU *****

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements are:

300

200

100

***** MENU *****

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 2

Deleted : 300

***** MENU *****

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 3

Stack elements are:

200

100

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#define SIZE 10
void enQueue(int);
void deQueue();
void display();
int queue[SIZE], front = -1, rear = -1;
void main()
{
 int value, choice;
 clrscr();
 while(1){
 printf("\n\n***** MENU *****\n");
 printf("1. Insertion\n2. Deletion\n3. Display\n4. Exit");
 printf("\nEnter your choice: ");
 scanf("%d",&choice);
 switch(choice){
 case 1: printf("Enter the value to be insert: ");
 scanf("%d",&value);
 enQueue(value);
 break;
 case 2: deQueue();
 break;
 case 3: display();
 break;
 case 4: exit(0);
 default: printf("\nWrong selection!!! Try again!!!");
 }
 }
}
void enQueue(int value){
 if(rear == SIZE-1)
 printf("\nQueue is Full!!! Insertion is not possible!!!");
 else{
 if(front == -1)
 front = 0;
 rear++;
 queue[rear] = value;
 printf("\nInsertion success!!!");
 }
}
void deQueue(){
 if(front == rear)
 printf("\nQueue is Empty!!! Deletion is not possible!!!");
 else{
 printf("\nDeleted : %d", queue[front]);
 front++;
 if(front == rear)
```

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

OUTPUT

```
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 15
Insertion success!!!
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 25
Insertion success!!!
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 35
Insertion success!!!
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 3
Queue elements are:
15 25 35
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 2
Deleted : 15
***** MENU *****
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice: 3
Queue elements are:
25  35
```

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include <conio.h>
#include<stdlib.h>
#define MAX 10
void create();
void insert();
void deletion();
void search();
void display();
int a,b[20], n, p, e, f, i, pos,flag=0;
void main()
{
//clrscr();
int ch;
char g='y';
do
{
printf("\n Main Menu");
printf("\n 1.Create \n 2.Delete \n 3.Search \n 4.Insert \n 5.Display\n 6.Exit \n");
printf("\n Enter your Choice");
scanf("%d", &ch);
switch(ch)
{
case 1:
create();
break;
case 2:
deletion();
break;
case 3:
search();
break;
case 4:
insert();
break;
case 5:
display();
break;
case 6:
exit(0);
break;
default:
printf("\n Enter the correct choice:");
}
printf("\n Do u want to continue:::");
scanf("\n%c", &g);
}
while(g=='y'||g=='Y');
```

```
getch();
}
void create()
{
printf("\n Enter the number of nodes");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n Enter the Element:",i+1);
scanf("%d", &b[i]);
}
}
void deletion()
{
printf("\n Enter the position u want to delete:");
scanf("%d", &pos);
if(pos>=n)
{
printf("\n Invalid Location::");
}
else
{
for(i=pos+1;i<n;i++)
{
b[i-1]=b[i];
}
n--;
}
printf("\n The Elements after deletion");
for(i=0;i<n;i++)
{
printf("\t%d", b[i]);
}
}
void search()
{
printf("\n Enter the Element to be searched:");
scanf("%d", &e);
for(i=0;i<n;i++)
{
if(b[i]==e)
{
 flag=1;
}
else
{
 flag=0;
 continue;
}
if(flag==1)
```

```
{
printf("Value is in the %d Position", i);
}
else
{
printf("Value %d is not in the list::", e);
}
}
}
void insert()
{
printf("\n Enter the position u need to insert::");
scanf("%d", &pos);
if(pos>=n)
 {
 printf("\n invalid Location::");
 }
 else
 {
 for(i=MAX-1;i>=pos-1;i--)
 {
 b[i+1]=b[i];
 }
 printf("\n Enter the element to insert::\n");
 scanf("%d",&p);
 b[pos]=p;
 n++;
 }
 printf("\n The list after insertion::\n");
display();
}
void display()
{
printf("\n The Elements of The list ADT are:");
for(i=0;i<n;i++)
{
printf("\n\n%d", b[i]);
}
}
```

OUTPUT

```
Main Menu
 1.Create
 2.Delete
 3.Search
 4.Insert
 5.Display
 6.Exit

Enter your Choice1
Enter the number of nodes3
Enter the Element:10
Enter the Element:20
Enter the Element:30
Do u want to continue:::y
main Menu
 1.Create
 2.Delete
 3.Search
 4.Insert
 5.Display
 6.Exit

 Enter your Choice4
 Enter the position u need to insert::2
 Enter the element to insert::
25
The list after insertion::
 The Elements of The list ADT are:
10
20
25
30
 Do u want to continue:::y
 main Menu
 1.Create
 2.Delete
 3.Search
 4.Insert
 5.Display
 6.Exit
 Enter your Choice3

 Enter the Element to be searched:30
 Value is in the 3 Position
 Do u want to continue:::y
 main Menu
 1.Create
 2.Delete
```

3.Search
4.Insert
5.Display
6.Exit

Enter your Choice2
Enter the position u want to delete::1
The Elements after deletion 10 25 30

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
struct Node
{
 int data;
 struct Node *next;
}*top = NULL;
void push(int);
void pop();
void display();
void main()
{
 int choice, value;
 clrscr();
 printf("\n:: Stack using Linked List ::\n");
 while(1){
 printf("\n****** MENU ******\n");
 printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");
 printf("Enter your choice: ");
 scanf("%d",&choice);
 switch(choice){
 case 1: printf("Enter the value to be insert: ");
 scanf("%d", &value);
 push(value);
 break;
 case 2: pop(); break;
 case 3: display(); break;
 case 4: exit(0);
 default: printf("\nWrong selection!!! Please try again!!!\n");
 }
 }
}
void push(int value)
{
 struct Node *newNode;
 newNode = (struct Node*)malloc(sizeof(struct Node));
 newNode->data = value;
 if(top == NULL)
 newNode->next = NULL;
 else
 newNode->next = top;
 top = newNode;
 printf("\nInsertion is Success!!!\n");
}
void pop()
{
 if(top == NULL)
 printf("\nStack is Empty!!!\n");
```

```
 else{
 struct Node *temp = top;
 printf("\nDeleted element: %d", temp->data);
 top = temp->next;
 free(temp);
 }
}
void display()
{
 if(top == NULL)
 printf("\nStack is Empty!!!\n");
 else{
 struct Node *temp = top;
 while(temp->next != NULL){
 printf("%d--->",temp->data);
 temp = temp -> next;
 }
 printf("%d--->NULL",temp->data);
 }
```

OUTPUT

```
:: Stack using Linked List ::
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 50
Insertion is Success!!!
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 60
Insertion is Success!!!
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 70
Insertion is Success!!!
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
70--->60--->50--->NULL
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Deleted element: 70
****** MENU ******
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
60--->50--->NULL
```

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
struct Node
{
 int data;
 struct Node *next;
}*front = NULL,*rear = NULL;
void insert(int);
void delete();
void display();
void main()
{
 int choice, value;
 clrscr();
 printf("\n:: Queue Implementation using Linked List ::\n");
 while(1){
 printf("\n****** MENU ******\n");
 printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
 printf("Enter your choice: ");
 scanf("%d",&choice);
 switch(choice){
 case 1: printf("Enter the value to be insert: ");
 scanf("%d", &value);
 insert(value);
 break;
 case 2: delete(); break;
 case 3: display(); break;
 case 4: exit(0);
 default: printf("\nWrong selection!!! Please try again!!!\n");
 }
 }
}
void insert(int value)
{
 struct Node *newNode;
 newNode = (struct Node*)malloc(sizeof(struct Node));
 newNode->data = value;
 newNode -> next = NULL;
 if(front == NULL)
 front = rear = newNode;
 else{
 rear -> next = newNode;
 rear = newNode;
 }
 printf("\nInsertion is Success!!!\n");
}
void delete()
{
```

```
 if(front == NULL)
 printf("\nQueue is Empty!!!\n");
 else{
 struct Node *temp = front;
 front = front -> next;
 printf("\nDeleted element: %d\n", temp->data);
 free(temp);
 }
}
void display()
{
 if(front == NULL)
 printf("\nQueue is Empty!!!\n");
 else{
 struct Node *temp = front;
 while(temp->next != NULL){
 printf("%d--->",temp->data);
 temp = temp -> next;
 }
 printf("%d--->NULL\n",temp->data);
 }
}
```

OUTPUT

:: Queue Implementation using Linked List ::

****** MENU ******

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 1

Enter the value to be insert: 65

Insertion is Success!!!

****** MENU ******

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 1

Enter the value to be insert: 75

Insertion is Success!!!

****** MENU ******

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 1

Enter the value to be insert: 85

Insertion is Success!!!

****** MENU ******

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 3

65--->75--->85--->NULL

****** MENU ******

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 2

Deleted element: 65

****** MENU ******

1. Insert

2. Delete

3. Display

4. Exit

Enter your choice: 3

75--->85--->NULL

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
void insertAtBeginning(int);
void insertAtEnd(int);
void insertAtAfter(int,int);
void deleteBeginning();
void deleteEnd();
void deleteSpecific(int);
void display();
struct Node
{
 int data;
 struct Node *previous, *next;
}*head = NULL;
void main()
{
 int choice1, choice2, value, location;
 clrscr();
 while(1)
 {
 printf("\n********** MENU **********\n");
 printf("1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");
 scanf("%d",&choice1);
 switch(choice1)
 {
 case 1: printf("Enter the value to be inserted: ");
 scanf("%d",&value);
 while(1)
 {
 printf("\nSelect from the following Inserting options\n");
 printf("1. At Beginning\n2. At End\n3. After a Node\n4. Cancel\nEnter your choice: ");
 scanf("%d",&choice2);
 switch(choice2)
 {
 case 1: insertAtBeginning(value);
 break;
 case 2: insertAtEnd(value);
 break;
 case 3: printf("Enter the location after which you want to insert: ");
 scanf("%d",&location);
 insertAtAfter(value,location);
 break;
 case 4: goto EndSwitch;
 default: printf("\nPlease select correct Inserting option!!!\n");
 }
 }
 case 2: while(1)
 {
```

```
 printf("\nSelect from the following Deleting options\n");
 printf("1. At Beginning\n2. At End\n3. Specific Node\n4. Cancel\nEnter your choice:
");
 scanf("%d",&choice2);
 switch(choice2)
 {
 case 1: deleteBeginning();
 break;
 case 2: deleteEnd();
 break;
 case 3: printf("Enter the Node value to be deleted: ");
 scanf("%d",&location);
 deleteSpecific(location);
 break;
 case 4: goto EndSwitch;
 default: printf("\nPlease select correct Deleting option!!!\n");
 }
 }
 EndSwitch: break;
 case 3: display();
 break;
 case 4: break;
 default: printf("\nPlease select correct option!!!");
 }
 }
}
void insertAtBeginning(int value)
{
 struct Node *newNode;
 newNode = (struct Node*)malloc(sizeof(struct Node));
 newNode -> data = value;
 newNode -> previous = NULL;
 if(head == NULL)
 {
 newNode -> next = NULL;
 head = newNode;
 }
 else
 {
 newNode -> next = head;
 head = newNode;
 }
 printf("\nInsertion success!!!");
}
void insertAtEnd(int value)
{
 struct Node *newNode;
 newNode = (struct Node*)malloc(sizeof(struct Node));
 newNode -> data = value;
 newNode -> next = NULL;
```

```
 if(head == NULL)
 {
 newNode -> previous = NULL;
 head = newNode;
 }
 else
 {
 struct Node *temp = head;
 while(temp -> next != NULL)
 temp = temp -> next;
 temp -> next = newNode;
 newNode -> previous = temp;
 }
 printf("\nInsertion success!!!");
}
void insertAtAfter(int value, int location)
{
 struct Node *newNode;
 newNode = (struct Node*)malloc(sizeof(struct Node));
 newNode -> data = value;
 if(head == NULL)
 {
 newNode -> previous = newNode -> next = NULL;
 head = newNode;
 }
 else
 {
 struct Node *temp1 = head, *temp2;
 while(temp1 -> data != location)
 {
 if(temp1 -> next == NULL)
 {
 printf("Given node is not found in the list!!!");
 goto EndFunction;
 }
 else
 {
 temp1 = temp1 -> next;
 }
 }
 temp2 = temp1 -> next;
 temp1 -> next = newNode;
 newNode -> previous = temp1;
 newNode -> next = temp2;
 temp2 -> previous = newNode;
 printf("\nInsertion success!!!");
 }
 EndFunction:
}
void deleteBeginning()
```

```
{
 if(head == NULL)
 printf("List is Empty!!! Deletion not possible!!!");
 else
 {
 struct Node *temp = head;
 if(temp -> previous == temp -> next)
 {
 head = NULL;
 free(temp);
 }
 else{
 head = temp -> next;
 head -> previous = NULL;
 free(temp);
 }
 printf("\nDeletion success!!!");
 }
}
void deleteEnd()
{
 if(head == NULL)
 printf("List is Empty!!! Deletion not possible!!!");
 else
 {
 struct Node *temp = head;
 if(temp -> previous == temp -> next)
 {
 head = NULL;
 free(temp);
 }
 else{
 while(temp -> next != NULL)
 temp = temp -> next;
 temp -> previous -> next = NULL;
 free(temp);
 }
 printf("\nDeletion success!!!");
 }
}
void deleteSpecific(int delValue)
{
 if(head == NULL)
 printf("List is Empty!!! Deletion not possible!!!");
 else
 {
 struct Node *temp = head;
 while(temp -> data != delValue)
 {
 if(temp -> next == NULL)
```

```
 {
 printf("\nGiven node is not found in the list!!!");
 goto FuctionEnd;
 }
 else
 {
 temp = temp -> next;
 }
 }
 if(temp == head)
 {
 head = NULL;
 free(temp);
 }
 else
 {
 temp -> previous -> next = temp -> next;
 free(temp);
 }
 printf("\nDeletion success!!!");
 }
 FuctionEnd:
}
void display()
{
 if(head == NULL)
 printf("\nList is Empty!!!");
 else
 {
 struct Node *temp = head;
 printf("\nList elements are: \n");
 printf("NULL <--- ");
 while(temp -> next != NULL)
 {
 printf("%d <===> ",temp -> data);
 temp=temp->next;
 }
 printf("%d ---> NULL", temp -> data);
 }
}
```

OUTPUT

********** MENU **********
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 10
Select from the following Inserting options
1. At Beginning
2. At End
3. After a Node
4. Cancel
Enter your choice: 1
Insertion success!!!
Select from the following Inserting options
1. At Beginning
2. At End
3. After a Node
4. Cancel
Enter your choice: 4

********** MENU **********
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be inserted: 20
Select from the following Inserting options
1. At Beginning
2. At End
3. After a Node
4. Cancel
Enter your choice: 2
Insertion success!!!
Select from the following Inserting options
1. At Beginning
2. At End
3. After a Node
4. Cancel
Enter your choice: 4

********** MENU **********
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1

```
Enter the value to be inserted: 25
Select from the following Inserting options
1. At Beginning
2. At End
3. After a Node
4. Cancel
Enter your choice: 3
Enter the location after which you want to insert: 10
Insertion success!!!
Select from the following Inserting options
1. At Beginning
2. At End
3. After a Node
4. Cancel
Enter your choice: 4

********** MENU **********
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
List elements are:
NULL <--- 10 <===> 25 <===> 20 ---> NULL

********** MENU **********
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Select from the following Deleting options
1. At Beginning
2. At End
3. Specific Node
4. Cancel
Enter your choice: 2
Deletion success!!!
Select from the following Deleting options
1. At Beginning
2. At End
3. Specific Node
4. Cancel
Enter your choice: 3
Enter the Node value to be deleted: 25
Deletion success!!!
Select from the following Deleting options
1. At Beginning
2. At End
3. Specific Node
```

4. Cancel
Enter your choice: 4

********** MENU **********
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
List elements are:
NULL <--- 10 ---> NULL

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#define SIZE 100
void enQueue(int);
int deQueueFront();
int deQueueRear();
void enQueueRear(int);
void enQueueFront(int);
void display();
int queue[SIZE];
int rear = 0, front = 0;
int main()
{
 char ch;
 int choice1, choice2, value;
 printf("\n******* Type of Double Ended Queue *******\n");
 do
 {
 printf("\n1.Input-restricted deque \n");
 printf("2.output-restricted deque \n");
 printf("\nEnter your choice of Queue Type : ");
 scanf("%d",&choice1);
 switch(choice1)
 {
 case 1:
 printf("\nSelect the Operation\n");
 printf("1.Insert\n2.Delete from Rear\n3.Delete from Front\n4. Display");
 do
 {
 printf("\nEnter your choice for the operation in c deque: ");
 scanf("%d",&choice2);
 switch(choice2)
 {
 case 1: enQueueRear(value);
 display();
 break;
 case 2: value = deQueueRear();
 printf("\nThe value deleted is %d",value);
 display();
 break;
 case 3: value=deQueueFront();
 printf("\nThe value deleted is %d",value);
 display();
 break;
 case 4: display();
 break;
 default:printf("Wrong choice");
 }
 printf("\nDo you want to perform another operation (Y/N): ");
```

```
 ch=getch();
 }while(ch=='y'||ch=='Y');
 getch();
 break;

 case 2 :
 printf("\n---- Select the Operation ----\n");
 printf("1. Insert at Rear\n2. Insert at Front\n3. Delete\n4. Display");
 do
 {
 printf("\nEnter your choice for the operation: ");
 scanf("%d",&choice2);
 switch(choice2)
 {
 case 1: enQueueRear(value);
 display();
 break;
 case 2: enQueueFront(value);
 display();
 break;
 case 3: value = deQueueFront();
 printf("\nThe value deleted is %d",value);
 display();
 break;
 case 4: display();
 break;
 default:printf("Wrong choice");
 }
 printf("\nDo you want to perform another operation (Y/N): ");
 ch=getch();
 } while(ch=='y'||ch=='Y');
 getch();
 break ;
 }
 printf("\nDo you want to continue(y/n):");
 ch=getch();
 }while(ch=='y'||ch=='Y');
}
void enQueueRear(int value)
{
 char ch;
 if(front == SIZE/2)
 {
 printf("\nQueue is full!!! Insertion is not possible!!! ");
 return;
 }
 do
 {
 printf("\nEnter the value to be inserted:");
 scanf("%d",&value);
```

```
 queue[front] = value;
 front++;
 printf("Do you want to continue insertion Y/N");
 ch=getch();
 }while(ch=='y');
}
void enQueueFront(int value)
{
 char ch;
 if(front==SIZE/2)
 {
 printf("\nQueue is full!!! Insertion is not possible!!!");
 return;
 }
 do
 {
 printf("\nEnter the value to be inserted:");
 scanf("%d",&value);
 rear--;
 queue[rear] = value;
 printf("Do you want to continue insertion Y/N");
 ch = getch();
 }
 while(ch == 'y');
}
int deQueueRear()
{
 int deleted;
 if(front == rear)
 {
 printf("\nQueue is Empty!!! Deletion is not possible!!!");
 return 0;
 }
 front--;
 deleted = queue[front+1];
 return deleted;
}
int deQueueFront()
{
 int deleted;
 if(front == rear)
 {
 printf("\nQueue is Empty!!! Deletion is not possible!!!");
 return 0;
 }
 rear++;
 deleted = queue[rear-1];
 return deleted;
}
void display()
```

```
{
 int i;
 if(front == rear)
 printf("\nQueue is Empty!!! Deletion is not possible!!!");
 else{
 printf("\nThe Queue elements are:");
 for(i=rear; i < front; i++)
 {
 printf("%d\t ",queue[i]);
 }
 }
}
```

OUTPUT

******* Type of Double Ended Queue *******
1.Input-restricted deque
2.output-restricted deque
Enter your choice of Queue Type : 1
Select the Operation
1.Insert
2.Delete from Rear
3.Delete from Front
4. Display

Enter your choice for the operation in c deque: 1
Enter the value to be inserted:10

Do you want to continue insertion Y/N
The Queue elements are:10

Do you want to perform another operation (Y/N): y
Enter your choice for the operation in c deque: 1
Enter the value to be inserted:20

Do you want to continue insertion Y/N
The Queue elements are:10 20

Do you want to perform another operation (Y/N): y
Enter your choice for the operation in c deque: 1
Enter the value to be inserted:30

Do you want to continue insertion Y/N
The Queue elements are:10 20 30

Do you want to perform another operation (Y/N): y
Enter your choice for the operation in c deque: 2
The value deleted is 30
The Queue elements are:10 20

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
struct node {
 int data; //node will store some data
 struct node *right_child;
 struct node *left_child;
};

struct node* new_node(int x) {
 struct node *temp;
 temp = malloc(sizeof(struct node));
 temp -> data = x;
 temp -> left_child = NULL;
 temp -> right_child = NULL;
 return temp;
}

struct node* search(struct node * root, int x) {
 if (root == NULL || root -> data == x) //if root->data is x then the element is found
 return root;
 else if (x > root -> data) // x is greater, so we will search the right subtree
 return search(root -> right_child, x);
 else //x is smaller than the data, so we will search the left subtree
 return search(root -> left_child, x);
}

struct node* insert(struct node * root, int x) {
 //searching for the place to insert
 if (root == NULL)
 return new_node(x);
 else if (x > root -> data) // x is greater. Should be inserted to the right
 root -> right_child = insert(root -> right_child, x);
 else // x is smaller and should be inserted to left
 root -> left_child = insert(root -> left_child, x);
 return root;
}

struct node* find_minimum(struct node * root) {
 if (root == NULL)
 return NULL;
 else if (root -> left_child != NULL) // node with minimum value will have no left child
 return find_minimum(root -> left_child); // left most element will be minimum
 return root;
}

struct node* delete(struct node * root, int x) {
 if (root == NULL)
 return NULL;
```

```
 if (x > root -> data)
 root -> right_child = delete(root -> right_child, x);
 else if (x < root -> data)
 root -> left_child = delete(root -> left_child, x);
 else {

if (root -> left_child == NULL && root -> right_child == NULL) {
 free(root);
 return NULL;
 }

else if (root -> left_child == NULL || root -> right_child == NULL) {
 struct node *temp;
 if (root -> left_child == NULL)
 temp = root -> right_child;
 else
 temp = root -> left_child;
 free(root);
 return temp;
 }

else {
 struct node *temp = find_minimum(root -> right_child);
 root -> data = temp -> data;
 root -> right_child = delete(root -> right_child, temp -> data);
 }
 }
 return root;
}

void inorder(struct node *root) {
 if (root != NULL) // checking if the root is not null
 {
 inorder(root -> left_child); // traversing left child
 printf(" %d ", root -> data); // printing data at root
 inorder(root -> right_child); // traversing right child
 }
}
int main() {
 struct node *root;
 root = new_node(20);
 insert(root, 5);
 insert(root, 1);
 insert(root, 15);
 insert(root, 9);
 insert(root, 7);
 insert(root, 12);
 insert(root, 30);
 insert(root, 25);
 insert(root, 40);
```

```
 insert(root, 45);
 insert(root, 42);
 inorder(root);
 printf("\n");
 root = delete(root, 1);
 root = delete(root, 40);
 root = delete(root, 45);
 root = delete(root, 9);
 inorder(root);
 printf("\n");
 return 0;
}
```

OUTPUT

1 5 7 9 12 15 20 25 30 40 42 45
5 7 12 15 20 25 30 42

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
struct node {
 int data;

 struct node *leftChild;
 struct node *rightChild;
};
struct node *root = NULL;
void insert(int data) {
 struct node tempNode = (struct node) malloc(sizeof(struct node));
 struct node *current;
 struct node *parent;
 tempNode->data = data;
 tempNode->leftChild = NULL;
 tempNode->rightChild = NULL;
 //if tree is empty
 if(root == NULL) {
 root = tempNode;
 } else {
 current = root;
 parent = NULL;
 while(1) {
 parent = current;

 //go to left of the tree
 if(data < parent->data) {
 current = current->leftChild;

 //insert to the left
 if(current == NULL) {
 parent->leftChild = tempNode;
 return;
 }
 } //go to right of the tree
 else {
 current = current->rightChild;
 //insert to the right
 if(current == NULL) {
 parent->rightChild = tempNode;
 return;
 }
 }
 }
 }
}
struct node* search(int data) {
```

```
 struct node *current = root;
 printf("Visiting elements: ");
 while(current->data != data) {
 if(current != NULL)
 printf("%d ",current->data);
 //go to left tree
 if(current->data > data) {
 current = current->leftChild;
 }
 //else go to right tree
 else {
 current = current->rightChild;
 }
 //not found
 if(current == NULL) {
 return NULL;
 }
 }

 return current;
}
void pre_order_traversal(struct node* root) {
 if(root != NULL) {
 printf("%d ",root->data);
 pre_order_traversal(root->leftChild);
 pre_order_traversal(root->rightChild);
 }
}
void inorder_traversal(struct node* root) {
 if(root != NULL) {
 inorder_traversal(root->leftChild);
 printf("%d ",root->data);
 inorder_traversal(root->rightChild);
 }
}
void post_order_traversal(struct node* root) {
 if(root != NULL) {
 post_order_traversal(root->leftChild);
 post_order_traversal(root->rightChild);
 printf("%d ", root->data);
 }
}
int main() {
 int i;
 int array[7] = { 27, 14, 35, 10, 19, 31, 42 };
 for(i = 0; i < 7; i++)
 insert(array[i]);
 i = 31;
 struct node * temp = search(i);
 if(temp != NULL) {
```

```
 printf("[%d] Element found.", temp->data);
 printf("\n");
 }else {
 printf("[ x ] Element not found (%d).\n", i);
 }
 i = 15;
 temp = search(i);
 if(temp != NULL) {
 printf("[%d] Element found.", temp->data);
 printf("\n");
 }else {
 printf("[ x ] Element not found (%d).\n", i);
 }
 printf("\nPreorder traversal: ");
 pre_order_traversal(root);
 printf("\nInorder traversal: ");
 inorder_traversal(root);
 printf("\nPost order traversal: ");
 post_order_traversal(root);
 return 0;
}
```

OUTPUT

Visiting elements: 27 35 [31] Element found.
Visiting elements: 27 14 19 [x] Element not found.
[15].
Preorder traversal:27 14 10 19 35 31 42
Inorder traversal:10 14 19 27 31 35 42
Postorder traversal:10 19 14 31 42 35 27

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
 int key;
 struct Node *left;
 struct Node *right;
 int height;
};
int max(int a, int b);
// Calculate height
int height(struct Node *N) {
 if (N == NULL)
 return 0;
 return N->height;
}
int max(int a, int b) {
 return (a > b) ? a : b;
}
struct Node *newNode(int key) {
 struct Node *node = (struct Node *)
 malloc(sizeof(struct Node));
 node->key = key;
 node->left = NULL;
 node->right = NULL;
 node->height = 1;
 return (node);
}
struct Node *rightRotate(struct Node *y) {
 struct Node *x = y->left;
 struct Node *T2 = x->right;
 x->right = y;
 y->left = T2;
 y->height = max(height(y->left), height(y->right)) + 1;
 x->height = max(height(x->left), height(x->right)) + 1;
 return x;
}
struct Node *leftRotate(struct Node *x) {
 struct Node *y = x->right;
 struct Node *T2 = y->left;
 y->left = x;
 x->right = T2;
 x->height = max(height(x->left), height(x->right)) + 1;
 y->height = max(height(y->left), height(y->right)) + 1;
 return y;
}
int getBalance(struct Node *N) {
 if (N == NULL)
 return 0;
```

```
 return height(N->left) - height(N->right);
}
struct Node *insertNode(struct Node *node, int key) {
if (node == NULL)
 return (newNode(key));
 if (key < node->key)
 node->left = insertNode(node->left, key);
 else if (key > node->key)
 node->right = insertNode(node->right, key);
 else
 return node;
node->height = 1 + max(height(node->left),
 height(node->right));
 int balance = getBalance(node);
 if (balance > 1 && key < node->left->key)
 return rightRotate(node);
 if (balance < -1 && key > node->right->key)
 return leftRotate(node);
 if (balance > 1 && key > node->left->key) {
 node->left = leftRotate(node->left);
 return rightRotate(node);
 }
 if (balance < -1 && key < node->right->key) {
 node->right = rightRotate(node->right);
 return leftRotate(node);
 }
 return node;
}
struct Node *minValueNode(struct Node *node) {
 struct Node *current = node;
 while (current->left != NULL)
 current = current->left;
 return current;
}
struct Node *deleteNode(struct Node *root, int key) {
 // Find the node and delete it
 if (root == NULL)
 return root;
 if (key < root->key)
 root->left = deleteNode(root->left, key);
 else if (key > root->key)
 root->right = deleteNode(root->right, key);
 else {
 if ((root->left == NULL) || (root->right == NULL)) {
 struct Node *temp = root->left ? root->left : root->right;
 if (temp == NULL) {
 temp = root;
 root = NULL;
 } else
 *root = *temp;
```

```
 free(temp);
 } else {
 struct Node *temp = minValueNode(root->right);
 root->key = temp->key;
 root->right = deleteNode(root->right, temp->key);
 }
 }
 if (root == NULL)
 return root;
root->height = 1 + max(height(root->left),
 height(root->right));
 int balance = getBalance(root);
 if (balance > 1 && getBalance(root->left) >= 0)
 return rightRotate(root);
 if (balance > 1 && getBalance(root->left) < 0) {
 root->left = leftRotate(root->left);
 return rightRotate(root);
 }
 if (balance < -1 && getBalance(root->right) <= 0)
 return leftRotate(root);
 if (balance < -1 && getBalance(root->right) > 0) {
 root->right = rightRotate(root->right);
 return leftRotate(root);
 }
 return root;
}
void printPreOrder(struct Node *root) {
 if (root != NULL) {
 printf("%d ", root->key);
 printPreOrder(root->left);
 printPreOrder(root->right);
 }
}
int main() {
 struct Node *root = NULL;
 root = insertNode(root, 2);
 root = insertNode(root, 1);
 root = insertNode(root, 7);
 root = insertNode(root, 4);
 root = insertNode(root, 5);
 root = insertNode(root, 3);
 root = insertNode(root, 8);
 printPreOrder(root);
 root = deleteNode(root, 3);
 printf("\nAfter deletion: ");
 printPreOrder(root);
 return 0;
}
```

OUTPUT

4 2 1 3 7 5 8
After deletion: 4 2 1 7 5 8

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v)
{
int i;
reach[v]=1;
for(i=1;i<=n;i++)
if(a[v][i] && !reach[i])
{
printf("\n %d->%d",v,i);
dfs(i);
}
}
void main()
{
int i,j,count=0;
clrscr();
printf("Depth First Search");
printf("\n Enter number of vertices:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
reach[i]=0;
for(j=1;j<=n;j++)
a[i][j]=0;
}
printf("\n Enter the adjacency matrix:\n");
for(i=1;i<=n;i++)
```

```
for(j=1;j<=n;j++)
scanf("%d",&a[i][j]);
dfs(1);
printf("\n");
for(i=1;i<=n;i++)
{
if(reach[i])
count++;
}
if(count==n)
printf("\n Graph is connected");
else
printf("\n Graph is not connected");
getch();
}
```

OUTPUT

Depth First Search

Enter the number of vertices: 3

Enter the adjacency matrix:

1 2 3

1 2 3

1 2 3

1->2

2->3

Graph is connected

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT

EX.NO :		DATE :

AIM :

ALGORITHM :

PROGRAM

```
#include <stdio.h>
#include <conio.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;
 clrscr();
 printf("\nBreadth First Search");
 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");
 getch();
 }
```

OUTPUT

```
Breadth First Search
Enter the number of vertices: 3
Enter graph data in matrix form:
1 2 3
1 2 3
1 2 3
Enter the starting vertex: 2
The node which are reachable are:
1   2   3
```

CONTENTS	MARKS ALLOTED	MARKS OBTAINED
AIM AND ALGORITHM	05	
PROGRAM AND EXECUTION	10	
VIVA-VOCE	10	
TOTAL	25	

RESULT