

CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY
FACULTY OF TECHNOLOGY AND ENGINEERING
Devang Patel Institute of Advance Technology & Research
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CE245 Data Structures & Algorithms

Semester: IV

Academic Year: 2019-20

PRACTICAL LIST

Sr. No.	AIM OF THE PRACTICAL	Date	Page No.	Remark
1.	Write a program to store roll numbers of student in array who attended training program in random order. Write function for a) Searching whether particular student attended training program or not using linear search and sentinel search. b) Searching whether particular student attended training program or not using binary search and Fibonacci search.			
2.	Mark purchased Books from books store of standard 1 to 7. He purchased 4 books for each standard (for std.1 books are 1.1,1.2,1.3,1.4 and for std. 2 books are 2.1,2.2,2.3,2.4 and so on..). When he reached home, he opens the bag and sees that all the books got mixed. So, how he will sort all the books, according to the standards and their preference in that particular standard. (ex. : preference in std. 1 is $1.1 < 1.2 < 1.3$ 2.1: SELECTION SORT that arranges in descending order 2.2: INSERTION SORT that arranges in ascending order			
3.	Implement a menu driven program that performs following sorting algorithms 3.1 QUICK SORT that arranges in ascending order 3.2 MERGE SORT that arranges in descending order			

4.	<p>Perform following programs using Stack data structure:</p> <p>4.1 Sometimes a program requires two stacks containing the same type of items. If the two stacks are stored in separate arrays. Then one stack might overflow while there was considerable unused space in the other. A neat way to avoid the problem is to put all the space in one array and let one stack grow from one end of the array and the other stack start at the other end and grow in opposite direction i.e., toward the first</p>			
	<p>stack, in this way, if one stack turns out to be large and the other small, then they will still both fit, and there will be no overflow until all the space is actually used. Declare a new structure type Double stack that includes the array and the two indices top A and top B, and write functions Push A, Push B, Pop A and Pop B to handle the two stacks with in one Double Stack.</p> <p>4.2 Implement Tower of Hanoi Problem using Recursion.</p>			
5.	<p>We are developing software for a call centre. When a client calls, his/her call should be stored until there is a free service representative to pick the call. Calls should be processed in the same order they are received. Select appropriate data structure to build call centre software system.</p>			
6.	<p>Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of Second, third and final year of department can be granted membership on request. Similarly, one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write a program to maintain club member 's information using singly linked list. Store student PRN and Name. Write functions to</p> <ul style="list-style-type: none"> a) Add and delete the members as well as president or even secretary. b) Compute total number of members of club c) Display members d) Display list in reverse order using recursion e) Two linked lists exist for two divisions. Concatenate two lists. 			

7.	<p>Write a program to implement Circular Queue with all operations. Check the queue contents and conditions with different combinations of insert and delete operations. Show the content of circular queue with front and rear pointer after each operation. Initially, the queue is empty. The size of the queue is 5. The sequence of operation given below:</p> <ul style="list-style-type: none"> • Insert 10, 50, 40, 80 • Delete • Insert 200, 70, 150 • Delete • Delete • Delete 			
8.	Implement the program Display Linked List in Reverse			
9.	<p>Implement a city database using a BST to store the database records. Each database record contains the name of the city (a string of arbitrary length) and the coordinates of the city expressed as integer x- and y-coordinates. The BST should be organized by city name. Your database should allow records to be inserted, deleted by name or coordinate, and searched by name or coordinate. Another operation that should be supported is to print all records within a given distance of a specified point. Collect runtime statistics for each operation. Which operations can be implemented reasonably efficiently (i.e., in $\Theta(\log n)$ time in the average case) using a BST? Can the database system be made more efficient by using one or more additional BSTs to organize the records by location?</p>			
10.	Write a program that enters vertices, edges of a Graph and display sequence of vertices to traverse the graph in Depth First Search method.			
11.	In an array of 20 elements, arrange 15 different values, which are generated randomly. Use hash function to generate key and linear probing to avoid collision. $H(x) = (x \bmod 18) + 2$. Write a program to input and display the final values of array.			

PRACTICAL 1

AIM:

Write a program to store roll numbers of student in array who attended training program in random order. Write function for a) Searching whether particular student attended training program or not using linear search and sentinel search. b) Searching whether particular student attended training program or not using binary search and Fibonacci search.

PROGRAM:

```
#include<iostream>
using namespace std;
class search
{
    int a[20],x,i,n,f,mid,key,q,p;
public:
    void getdata();
    void lsearch(int m[20],int n);
    void ssearch(int m[20],int n);
    int bsearch(int m[20],int l,int r,int b);
    int fsearch(int m[20],int n);
    void display();
};
void search::getdata()
{
    cout<<"\nEnter the no.s of rollno : ";
    cin>>n;
    for(i=0;i<n;i++)
        cin>>a[i];
}
void search::lsearch(int m[20],int n)
{
    f=1;
    cout<<"\nEnter the element to be searched : ";
    cin>>x;
    for(i=0;i<n;i++)
    {
        if(x==m[i])
        {
            cout<<"\nRoll no. found at location : "<<i+1;    f=0;
            break;
        }
    }
    if(f==1)
        { cout<<"\n Rollno not found";    }
}
void search::ssearch(int m[20],int n)
{
    m[n]=-1; i=0;
```

```

    cout<<"\nEnter the element to be searched : ";
    cin>>x;
    while((x!=m[i])&&(m[i]!=-1))
        i++;
    if(i<n)
    {   cout<<"\nRollno is found at location : "<<i+1;  }
    else
    {   cout<<"\n Roll no not found";           }

}
int search::bsearch(int m[20],int l,int r,int b)
{
    if(l<=r)
    {
        mid=(l+r)/2;
        if(b==m[mid])
            return mid;
        else if(b>m[mid])
            return bsearch(m,(mid+1),r,b);
        else if(b<m[mid])
            return bsearch(m,l,(mid-1),b);
    }
    return -1;
}
int search::fsearch(int m[20],int n)
{   int k,f[20]={0,1,1,2,3,5,8,13,21,44,65},flag=1;
    cout<<"\nEnter the element to be searched : ";
    cin>>key;
    k=0;
    while(flag)
    {   if(f[k]>=n&&f[k-1]<n)
        {   break;  }
        k++;
    }
    mid=n-f[k-2]+1;
    p=f[k-2];
    q=f[k-3];
    while(flag)
    {
        if(key<m[mid])
        {   if(q==0)
            return -1;
            else
            {   mid=mid-q;
                p=f[k-3];
                q=f[k-4];
            }
        }
    }
}

```

```
        }
    }
    if(key==m[mid])
    {    return mid;    }

    if(key>m[mid])
    {    if(p==1)
        return -1;
        else
        {    mid=mid+q;
            p=f[k-4];
            q=f[k-5];
        }
    }
}
}

int main()
{
    search s;
    int c,d,v[20],w,i,j,choice,temp;
    cout<<"Parth Vasoya \n 18DCS134 \n";
    char ch;
    cout<<"\n How Many students :";
    cin>>d;
    cout<<endl<<"Enter the students Rollno."<<endl;

    for(i=0;i<d;i++)
        cin>>v[i];

    do{
        cout<<"\n1. Linear search";
        cout<<"\n2. Sentinel search";
        cout<<"\n3. Binary search";
        cout<<"\n4. Fibonacci search";
        cout<<"\n Enter your choice : ";
        cin>>choice;
        switch(choice)
        {
            case 1:
                s.lsearch(v,d);
                break;
            case 2:
                s.ssearch(v,d);
                break;
            case 3:
                for(i=0;i<d;i++)
```

```
{
    for(j=i+1;j<d;j++)
    {
        if(v[i]>v[j])
        {
            temp=v[i];
            v[i]=v[j];
            v[j]=temp;
        }
    }
}
cout<<"\nSorted array is... \n";
for(i=0;i<d;i++)
    cout<<v[i]<<" ";
cout<<"\nEnter the roll no. to be searched : ";
cin>>c;
w=s.bsearch(v,0,(d-1),c);
if(w==-1)
{
    cout<<"\n Roll no. not found ";
}
else
{
    cout<<"\n Roll no. found at location : "<<w+1;
}
break;
case 4:
    for(i=0;i<d;i++)
    {
        for(j=i+1;j<d;j++)
        {
            if(v[i]>v[j])
            {
                temp=v[i];
                v[i]=v[j];
                v[j]=temp;
            }
        }
    }
    cout<<"\nSorted array is... \n";
    for(i=0;i<d;i++)
        cout<<v[i]<<" ";
    w=s.fsearch(v,d);
    if(w==-1)
    {
        cout<<"\n Roll no. not found ";
    }
```

```
    }
    else
    {
        cout<<"\n Roll no. found at location : "<<w+1;
    }
    break;
default:
    cout<<"\n Invalid choice";
}
cout<<"\n Do you want to continue (y/n): ";
cin>>ch;
}while(ch=='y'||ch=='Y');
return 0;
}
```

OUTPUT:

```
Parth Vasoya
18DCS134

How Many students :5

Enter the students Rollno.
134
100
125
107
110

1. Linear search
2. Sentinel search
3. Binary search
4. Fibonacci search
Enter your choice : 1

Enter the element to be searched : 134

Roll no. found at location : 1
Do you want to continue (y/n): y

1. Linear search
2. Sentinel search
3. Binary search
4. Fibonacci search
Enter your choice : 2

Enter the element to be searched : 100

Rollno is found at location : 2
Do you want to continue (y/n): y
```



```
Rollno is found at location : 2
Do you want to continue (y/n): y

1. Linear search
2. Sentinel search
3. Binary search
4. Fibonacci search
Enter your choice : 3

Sorted array is...
100 107 110 125 134
Enter the roll no. to be searched : 134

Roll no. found at location : 5
Do you want to continue (y/n): y

1. Linear search
2. Sentinel search
3. Binary search
4. Fibonacci search
Enter your choice : 4

Sorted array is...
100 107 110 125 134
Enter the element to be searched : 134

Roll no. found at location : 5
Do you want to continue (y/n): y

1. Linear search
2. Sentinel search
3. Binary search
4. Fibonacci search
Enter your choice : 1

Enter the element to be searched : 1222

Rollno not found
Do you want to continue (y/n):
```

CONCLUSION:

In this practical, we performed four types of searching algorithms which are linear search, binary search, sentinel search, Fibonacci search.

PRACTICAL 2

AIM:

Mark purchased Books from books store of standard 1 to 7. He purchased 4 books for each standard (for std.1 books are 1.1,1.2,1.3,1.4 and for std. 2 books are 2.1,2.2,2.3,2.4 and so on.). When he reached home, he opens the bag and sees that all the books got mixed. So, how he will sort all the books, according to the standards and their preference in that particular standard. (ex. : preference in std. 1 is $1.1 < 1.2 < 1.3$)

2.1: SELECTION SORT that arranges in descending order

2.2: INSERTION SORT that arranges in ascending order

2.1) SELECTION SORT

PROGRAM:

```
//Practical 2 selection sort descending order
#include<iostream>
using namespace std;
int main()
{
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    float
b[32]={ 1.2,1.3,1.1,1.4,2.1,2.4,2.2,2.3,3.2,3.3,3.1,3.4,4.1,4.4,4.2,4.3,5.2,5.3,5.1,5.4,7.1,7.4,7.2,7.3
,8.2,8.3,8.1,8.4,6.1,6.4,6.2,6.3};
    int i,j,k=0,index;
    float mini,temp;
    for( i=0;i<32;i++)
    {
        mini=b[i];
        index = i;
        for( j=i+1;j<32;j++)
        {
            if(mini>b[j])
            {
                index=j;
                mini=b[j];
            }
        }

        temp=b[index];
        b[index]=b[i];
        b[i]=temp;
    }
    cout<<endl<<endl<<"Sorted array"<<endl<<endl;
    for(int i=0;i<32;i++)
```

```

    {
        cout<<b[i]<<endl;
    }
}

```

OUTPUT:

```
Parth Vasoya
18DCS134

Sorted array

1.1
1.2
1.3
1.4
2.1
2.2
2.3
2.4
3.1
3.2
3.3
3.4
4.1
4.2
4.3
4.4
5.1
5.2
5.3
5.4
6.1
6.2
6.3

6.1
6.2
6.3
6.4
7.1
7.2
7.3
7.4
8.1
8.2
8.3
8.4

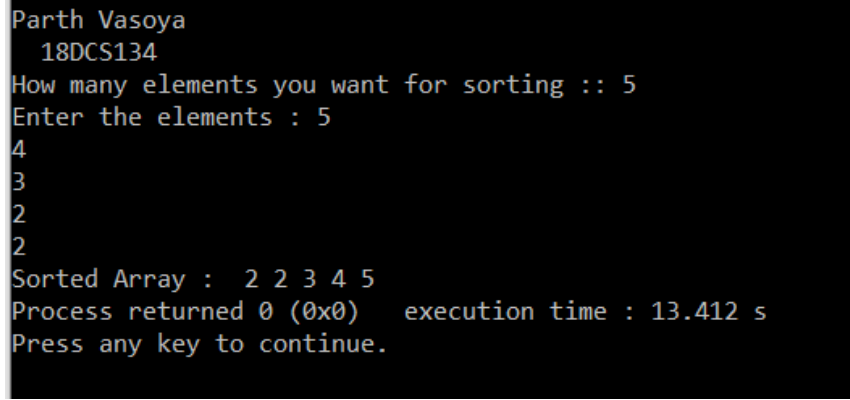
Process returned 0 (0x0)   execution time : 0.163 s
Press any key to continue.
```

2.2) INSERTION SORT

PROGRAM:

```
#include<iostream>
using namespace std;
int main()
{
    int a[30],i,j,key,n;
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    cout<<"How many elements you want for sorting :: ";
    cin>>n;
    cout<<"Enter the elements : ";
    for(int i =0;i<n;i++)
        cin>>a[i];
    for(int i =1;i<n;i++)
    {
        int key = a[i];
        int j=i-1;
        while((j>=0)&&(a[j]>key))
        {
            a[j+1]=a[j];
            j=j-1;
        }
        a[j+1]=key;
    }
    cout<<"Sorted Array : ";
    for(int i =0;i<n;i++)
        cout<<" "<<a[i];
}
```

OUTPUT:



```
Parth Vasoya
18DCS134
How many elements you want for sorting :: 5
Enter the elements : 5
4
3
2
2
Sorted Array : 2 2 3 4 5
Process returned 0 (0x0)   execution time : 13.412 s
Press any key to continue.
```

CONCLUSION:

In this practical, we performed two sorting algorithms : Selection sort and Insertion sort.

PRACTICAL 3

AIM:

Implement a menu driven program that performs following sorting algorithms

3.1 QUICK SORT that arranges in ascending order

3.2 MERGE SORT that arranges in descending order

3.1) QUICK SORT

PROGRAM:

```
#include<iostream>
using namespace std;
void swapping(int &a, int &b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
void display(int *array, int size) {
    for(int i = 0; i<size; i++)
        cout<< array[i] << " ";
    cout<<endl;
}
int partition(int *array, int lower, int upper)
{
    int pivot, start, end;
    pivot = array[lower];
    start = lower; end = upper;

    while(start < end) {
        while(array[start] <= pivot && start<end)
        {
            start++;
        }

        while(array[end] > pivot) {
            end--;
        }
        if(start < end) {
            swap(array[start], array[end]);
        }
    }
}
```

```
    array[lower] = array[end];
    array[end] = pivot;
    return end;
}

void quickSort(int *array, int left, int right) {
    int q;

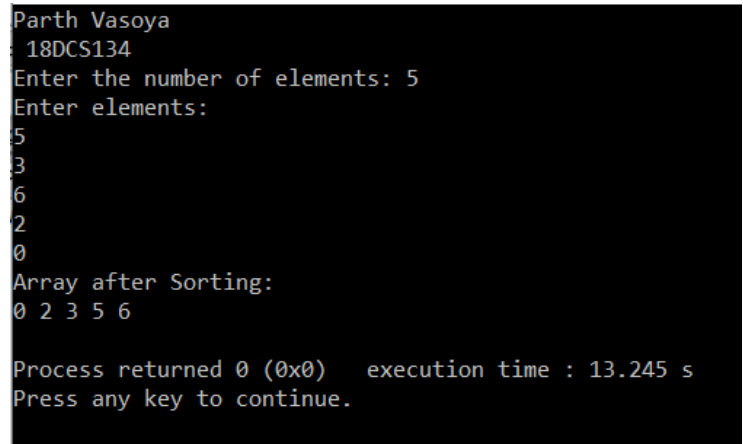
    if(left < right) {
        q = partition(array, left, right);
        quickSort(array, left, q-1);
        quickSort(array, q+1, right);
    }
}

int main() {
    cout<<"Parth Vasoya \n 18DCS134 \n";
    int n;
    cout<< "Enter the number of elements: ";
    cin>> n;
    int arr[n];
    cout<< "Enter elements:" <<endl;

    for(int i = 0; i<n; i++) {
        cin>>arr[i];
    }

    quickSort(arr, 0, n-1);
    cout<< "Array after Sorting: " <<endl;
    display(arr, n);
}
```

OUTPUT:



```
Parth Vasoya
18DCS134
Enter the number of elements: 5
Enter elements:
5
3
6
2
0
Array after Sorting:
0 2 3 5 6

Process returned 0 (0x0)   execution time : 13.245 s
Press any key to continue.
```

3.2) MERGE SORT:

PROGRAM:

```
#include<iostream>
using namespace std;
void Merge(int *a,int low,int ub,int mid)
{
    int i,j,k,b[ub-low+1];
    i=low;
    k=0;
    j=mid+1;

    while(i<=mid && j<=ub)
    {
        if(a[i]<a[j])
        {
            b[k]=a[i];
            k++;
            i++;
        }
        else
        {
            b[k]=a[j];
            k++;
            j++;
        }
    }
    while(i<=mid)
    {
        b[k]=a[i];
        k++;
        i++;
    }
    while(j<=ub)
    {
        b[k]=a[j];
        k++;
        j++;
    }

    for(i=low;i<=ub;i++)
    {
        a[i]=b[i-low];
    }
}
```

```
void MergeSort(int *a,int low,int ub)
{
    int mid;

    if(low<ub)
    {
        mid=(low+ub)/2;
        MergeSort(a,low,mid);
        MergeSort(a,mid+1,ub);
        Merge(a,low,ub,mid);
    }

}

int main()
{
    cout<<"Parth Vasoya \n 18DCS134 \n";
    int i,n;
    cout<<"\nEnter the number of data elements to be sorted :";
    cin>>n;

    int a[n];

    for(i=0;i<n;i++)
    {
        cout<<"Enter the element "<<i+1<<": ";
        cin>>a[i];
    }

    MergeSort(a, 0, n-1);
    cout<<"\nSorted Data : ";
    for (i = 0; i< n; i++)
        cout<<" "<<a[i];

    return 0;
}
```


OUTPUT:

```
Parth Vasoya
18DCS134

Enter the number of data elements to be sorted :5
Enter the element 1: 43
Enter the element 2: 64
Enter the element 3: 85
Enter the element 4: 23
Enter the element 5: 8

Sorted Data : 8 23 43 64 85
Process returned 0 (0x0)   execution time : 9.280 s
Press any key to continue.
```

CONCLUSION:

In this practical we learned about merge sort and quick sort .

PRACTICAL 4

AIM:

Perform following programs using Stack data structure:

4.1 Sometimes a program requires two stack containing the same type of items. If the two stacks are stored in separate arrays. Then one stack might overflow while there was considerable unused space in the other. A neat way to avoid the problem is to put all the space in one array and let one stack grow from one end of the array and the other stack start at the other end and grow in opposite direction i.e., toward the first stack, in this way, if one stack turns out to be large and the other small, then they will still both fit, and there will be no overflow until all the space is actually used. Declare a new structure type Double stack that includes the array and the two indices top A and top B, and write functions Push A, Push B, Pop A and Pop B to handle the two stacks with in one Double Stack.

4.2 Implement Tower of Hanoi Problem using Recursion.

4.1

PROGRAM:

```
#include <stdio.h>
#include <iostream>
#include <string.h>
#define size 5

using namespace std;

struct doubleStack
{
    int stack[size];
    int topA, topB;
};

void pushA(struct doubleStack &s, int d)
{
    s.stack[++(s.topA)] = d;
}

void pushB(struct doubleStack &s, int d)
{
    s.stack[--(s.topB)] = d;
}

int popA(struct doubleStack &s)
{
    int popped_element = s.stack[(s.topA)];
    (s.topA)--;
    return (popped_element);
}
```

```
}
int popB(struct doubleStack &s)
{
    int popped_element = s.stack[(s.topB)];
    (s.topB)++;
    return (popped_element);
}
int empty(struct doubleStack &s)
{
    if ((s.topA == -1) && (s.topB == 1))
        return 1;
    return 0;
}
int full(struct doubleStack &s)
{
    if ((s.topA + 1 == s.topB) || (s.topA >= size) || (s.topB <= 0))
        return 1;
    return 0;
}
void display(struct doubleStack &s)
{
    if (empty(s))
    {
        cout << "stack is empty";
    }
    else
    {
        cout << "\nStack B is :";
        for (int i = s.topB; i <= size - 1; ++i)
            cout << " " << s.stack[i];
        cout << "\nStack A is :";
        for (int j = s.topA; j >= 0; --j)
            cout << " " << s.stack[j];
    }
}
int main()
{
    struct doubleStack twoStack;
    int item;
    char choice;
    int q = 0;
    twoStack.topA = -1;
    twoStack.topB = size;
    cout << "Parth Vasoya \n 18DCS134\n";
    cout << " \n-----Double Stack-----";
    do
```

```
{
    cout << " \nPush to Stack A tap 1";
    cout << " \nPush to Stack B tap 2";
    cout << " \nPop to Stack A tap 3";
    cout << " \nPop to Stack B tap 4";
    cout << " \nTo Quit tap 5";

    cout << " \nInput the Choice.";
    do
    {
        cin >> choice;
    }

    while (strchr("11 12345 ", choice) == NULL);
    switch (choice)
    {
    case '1':
        cout << "\n Input the Element to be pushed:";
        cin >> item;
        if (!full(twoStack))
        {
            pushA(twoStack, item);
            cout << "\n After Inserting into Stack A.";
        }
        else
            cout << "\n Stack is now full.";
        break;
    case '2':
        cout << "\n Input the Element to be pushed:";
        cin >> item;
        if (!full(twoStack))
        {
            pushB(twoStack, item);
            cout << "\n After Inserting into Stack B.";
        }
        else
            cout << "\n Stack is now full.";
        break;
    case '3':
        if (!empty(twoStack))
        {
            item = popA(twoStack);
            cout << "\n Data is popped: " << item;
            cout << "\n Rest Data in Stack is as follows:\n";
        }
        else
```

```
        cout << "\n Stack Underflow.";
        break;
    case '4':
        if (!empty(twoStack))
        {
            item = popB(twoStack);
            cout << "\n Data is popped: " << item;
            cout << "\n Rest Data in Stack is as follows:\n";
        }
        else
            cout << "\n Stack Underflow.";
        break;
    case '5':
        q = 1;
    }
    display(twoStack);
} while (!q);
return 0;
}
```

OUTPUT:

```
Parth Vasoya
18DCS134

-----Double Stack-----
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.1

    Input the Element to be pushed:10

    After Inserting into Stack A.
Stack B is :
Stack A is : 10
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.1

    Input the Element to be pushed:30

    After Inserting into Stack A.
Stack B is :
Stack A is : 30 10
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.2

    Input the Element to be pushed:20

    After Inserting into Stack B.
Stack B is : 20
Stack A is : 30 10
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
```

```
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.2

    Input the Element to be pushed:40

    After Inserting into Stack B.
Stack B is : 40 20
Stack A is : 30 10
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.3

    Data is popped: 30
    Rest Data in Stack is as follows:

Stack B is : 40 20
Stack A is : 10
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.4

    Data is popped: 40
    Rest Data in Stack is as follows:

Stack B is : 20
Stack A is : 10
Push to Stack A tap 1
Push to Stack B tap 2
Pop to Stack A tap 3
Pop to Stack B tap 4
To Quit tap 5
Input the Choice.5

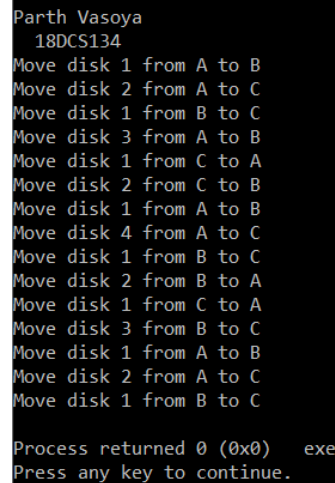
Stack B is : 20
Stack A is : 10
Process returned 0 (0x0) execution time
Press any key to continue.
```

4.2

PROGRAM:

```
#include <iostream>
using namespace std;
void towerOfHanoi(int n,char beg,char endd,char aux)
{
    if (n==1)
    {
        cout << "Move disk 1 from " << beg << " to " << endd << endl;
        return;
    }
    towerOfHanoi(n-1,beg,aux,ends);
    cout << "Move disk " << n << " from " << beg << " to " << endd << endl;
    towerOfHanoi(n-1,aux,ends,beg);
}
int main()
{
    int n = 4;
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    towerOfHanoi(n,'A','C','B');
    return 0;
}
```

OUTPUT:



```
Parth Vasoya
18DCS134
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C
Move disk 3 from A to B
Move disk 1 from C to A
Move disk 2 from C to B
Move disk 1 from A to B
Move disk 4 from A to C
Move disk 1 from B to C
Move disk 2 from B to A
Move disk 1 from C to A
Move disk 3 from B to C
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C

Process returned 0 (0x0)   exe
Press any key to continue.
```

CONCLUSION:

In this practical we learned about stack and tower of Hanoi using recursion.

PRACTICAL 5

AIM:

We are developing software for a call center. When a client calls, his/her call should be stored until there is a free service representative to pick the call. Calls should be processed in the same order they are received. Select appropriate data structure to build call center software system.

PROGRAM:

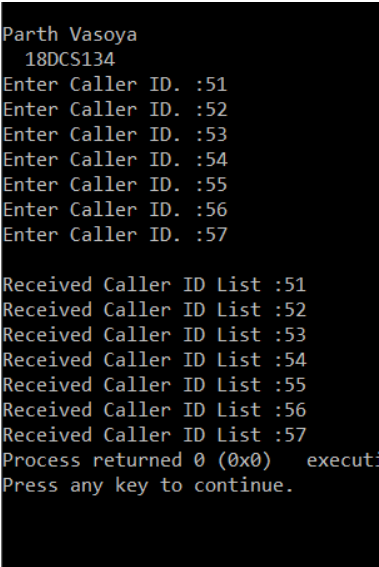
```
#include <iostream>
#define SIZE 7
using namespace std;

class Queue
{
    int a[SIZE];
    int rear;
    int front;
public:
    Queue()
    {
        rear = front = -1;
    }
    void enqueue(int x)
    {
        if (front == -1)
        {
            front++;
        }
        if (rear == SIZE - 1)
        {
            cout << "Queue is full";
        }
        else
        {
            a[++rear] = x;
        }
    }
    int dequeue()
    {
        cout << "\nReceived Caller ID List :" << a[+front];
        return a[++front];
    }
}
```



```
void display()
{
    int i;
    for (i = front; i <= rear; i++)
    {
        cout << a[i] << endl;
    }
}
};
int main()
{
    int x;
    Queue q;
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    for (int i = 0; i < SIZE; i++)
    {
        cout << "Enter Caller ID. :";
        cin >> x;
        q.enqueue(x);
    }

    for (int i = 0; i < SIZE; i++)
    {
        q.dequeue();
    }
}
```

OUTPUT:

```
Parth Vasoya
 18DCS134
Enter Caller ID. :51
Enter Caller ID. :52
Enter Caller ID. :53
Enter Caller ID. :54
Enter Caller ID. :55
Enter Caller ID. :56
Enter Caller ID. :57

Received Caller ID List :51
Received Caller ID List :52
Received Caller ID List :53
Received Caller ID List :54
Received Caller ID List :55
Received Caller ID List :56
Received Caller ID List :57
Process returned 0 (0x0)   execut
Press any key to continue.
```

CONCLUSION:

In this practical we have implemented queue using array.

PRACTICAL 6

AIM:

Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of Second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write a program to maintain club member 's information using singly linked list. Store student PRN and Name. Write functions to

- a) Add and delete the members as well as president or even secretary.
- b) Compute total number of members of club
- c) Display members
- d) Display list in reverse order using recursion
- e) Two linked lists exists for two divisions. Concatenate two lists.

PROGRAM:

```
#include <iostream>
#include <string>

using namespace std;

struct node
{
    string name;
    int prn;
    node *next;
};

class member
{
public:
    node *header1, *header2;
    member()
    {
        header1 = NULL;
        header2 = NULL;
    }

    node *create();
    int count(node *x);
    void add(node *head);
    void del(node *head);
    void display(node *head);
    void rdisplay(node *cn);
```

```
void concatenate();
};
int member::count(node *x)
{
    node *cn = x;
    int count = 0;
    while (cn != NULL)
    {
        count++;
        cn = cn->next;
    }
    return count;
}

node *member::create()
{
    char ch;

    node *head;
    node *nn = new node;
    head = nn;
    cout << "Enter Name of president : ";
    cin >> nn->name;
    cout << "Enter PRN of president : ";
    cin >> nn->prn;
    cout << endl;

    do
    {
        nn->next = new node;
        nn = nn->next;
        cout << "Enter Name of Member : ";
        cin >> nn->name;
        cout << "Enter PRN of Member : ";
        cin >> nn->prn;
        cout << endl;

        cout << "Do you want to enter another member? (Y or y if yes) : ";
        cin >> ch;
        cout << endl;

    } while (ch == 'Y' || ch == 'y');

    nn->next = new node;

    nn = nn->next;
```

```
    cout << "Enter Name of secretary : ";
    cin >> nn->name;
    cout << "Enter PRN of secretary : ";
    cin >> nn->prn;
    cout << endl;

    nn->next = NULL;
    cout << "List is created! It has " << count(head) << " members!" << endl
        << endl;

    return head;
}

void member::display(node *head)
{

    node *nn;
    nn = head;

    cout << "President : " << nn->name << endl;
    cout << "PRN : " << nn->prn << endl
        << endl;
    nn = nn->next;

    for (int i = 0; nn->next != NULL; i++)
    {
        cout << "Member : " << nn->name << endl;
        cout << "PRN : " << nn->prn << endl
            << endl;
        nn = nn->next;
    }

    cout << "Secretary : " << nn->name << endl;
    cout << "PRN : " << nn->prn << endl
        << endl;

    cout << "List has " << count(head) << " members!" << endl
        << endl;
}

void member::rdisplay(node *cn)
{
    if (cn == NULL)
        return;
```

```
    rdisplay(cn->next);
    cout << cn->name << endl;
    cout << cn->prn << endl;
    cout << endl;
}

void member::add(node *head)
{
    int p;
    cout << "Enter the position where you want to add : ";
    cin >> p;

    node *nn, *temp;
    nn = head;

    node *an = new node;
    if (p == 1)
    {
        cout << "Enter Name of Member : ";
        cin >> an->name;
        cout << "Enter PRN of Member : ";
        cin >> an->prn;
        cout << endl;
        an->next = head;
        head = an;
        cout << "Member is added! List has " << count(head) << " members now!\n\n";
    }
    else if (p == count(head) + 1)
    {
        for (int i = 0; nn->next != NULL; i++)
        {
            nn = nn->next;
        }

        cout << "Enter Name of Member : ";
        cin >> an->name;
        cout << "Enter PRN of Member : ";
        cin >> an->prn;
        cout << endl;
        nn->next = an;
        an->next = NULL;

        cout << "Member is added! List has " << count(head) << " members now!\n\n";
        ;
    }
    else if (1 < p && p <= count(head))
```

```
{
    for (int i = 1; i < p; i++)
    {
        temp = nn;
        nn = nn->next;
    }

    cout << "Enter Name of Member : ";
    cin >> an->name;
    cout << "Enter PRN of Member : ";
    cin >> an->prn;
    cout << endl;
    an->next = nn;
    temp->next = an;

    cout << "Member is added! List has " << count(head) << " members now!\n\n";
}
else
    cout << "Invalid Position!\n\n";
}

void member::del(node *head)
{
    int key;
    cout << "Enter the PRN of the student which is to be deleted : ";
    cin >> key;
    cout << endl;

    node *nn, *temp;
    nn = head;

    if (nn->prn == key)
    {
        head = nn->next;
        delete (nn);
        cout << "Member deleted! List has " << count(head) << " members now!\n\n";
    }
    else
    {
        while (nn->prn != key)
        {
            temp = nn;
            nn = nn->next;
        }

        if (nn->prn == key && nn->next == NULL)
```

```

    {
        temp->next = NULL;
        delete (nn);
        cout << "Member deleted! List has " << count(head) << " members now!\n\n";
    }
    else if (nn->prn == key)
    {
        temp->next = nn->next;
        delete (nn);
        cout << "Member deleted! List has " << count(head) << " members now!\n\n";
    }
    else
        cout << "Member not found!\n\n";
}
}

void member::concatinate()
{
    node *cn = header1;
    while (cn->next != NULL)
        cn = cn->next;

    cn->next = header2;

    cout << "Lists of Division A and Division B are concatenated! Club has total " << count(header1) << " members now!\n\n";
}

int main()
{
    int choice;
    member m;
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    cout << "-----WELCOME TO PINNACLE CLUB-----" << endl<< endl;
    cout << "Create A division list: " << endl<< endl;
    m.header1 = m.create();
    cout << "Create B division list: " << endl
        << endl;
    m.header2 = m.create();

    while (true)
    {
        cout << endl
            << "Enter 1 to add a member in division A\nEnter 2 to add a member in division B\nEnter 3 to delete a member from division A\nEnter 4 to delete a member from division B\nEnter 5 to display division A list\nEnter 6 to display division B list\nEnter 7 to reverse display division A l

```

ist\nEnter 8 to reverse display division B list\nEnter 9 to concatenate Division A and Division B\
nEnter 10 to display concatenated list\nEnter any other key to exit\nInput: ";

```
    cin >> choice;
    cout << endl;

    switch (choice)
    {

    case 1:
        m.add(m.header1);
        break;
    case 2:
        m.add(m.header2);
        break;
    case 3:
        m.del(m.header1);
        break;
    case 4:
        m.del(m.header2);
        break;
    case 5:
        m.display(m.header1);
        break;
    case 6:
        m.display(m.header2);
        break;
    case 7:
        m.rdisplay(m.header1);
        break;
    case 8:
        m.rdisplay(m.header2);
        break;
    case 9:
        m.concatenate();
        break;
    case 10:
        m.display(m.header1);
        break;
    default:
        return 0;
    }
}
```


OUTPUT:

```

Parth Vasoya
18DCS134
-----WELCOME TO PINNACLE CLUB-----

Create A division list:

Enter Name of president : Parth
Enter PRN of president : 134

Enter Name of Member : Bipin
Enter PRN of Member : 201

Do you want to enter another member? (Y or y if yes) : n

Enter Name of secretary : Nikul
Enter PRN of secretary : 301

List is created! It has 3 members!

Create B division list:

Enter Name of president :
Priyank
Enter PRN of president : 401

Enter Name of Member : Dev
Enter PRN of Member : 501

Do you want to enter another member? (Y or y if yes) : n

Enter Name of secretary : Nayan
Enter PRN of secretary : 601

List is created! It has 3 members!

Enter 1 to add a member in division A

```

```

List is created! It has 3 members!

Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Enter 7 to reverse display division A list
Enter 8 to reverse display division B list
Enter 9 to concatenate Division A and Division B
Enter 10 to display concatenated list
Esnter any other key to exit
Input: 1

Enter the position where you want to add : 4
Enter Name of Member : Chintan
Enter PRN of Member : 302

Member is added! List has 4 members now!

Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Enter 7 to reverse display division A list
Enter 8 to reverse display division B list
Enter 9 to concatenate Division A and Division B
Enter 10 to display concatenated list
Esnter any other key to exit
Input:

```

CONCLUSION:

In this practical we studied about linked list. Reverse of a linked list and traversal and deletion of element.

PRACTICAL 7**AIM:**

Write a program to implement Circular Queue with all operations. Check the queue contents and conditions with different combinations of insert and delete operations. Show the content of circular queue with front and rear pointer after each operation. Initially, the queue is empty. The size of the queue is 5. The sequence of operation given below:

Insert 10, 50, 40, 80

Delete

Insert 200, 70, 150

Delete

Delete

Delete

PROGRAM CODE:

```
#include <iostream>
using namespace std;

class Circular
{
private:
    int queue[5];
    int front = -1, rear = -1;

public:
    void insert(int x)
    {
        if (front == -1 && rear == -1)
        {
            queue[0] = x;
            front++;
            rear++;
        }
        else if (front == rear + 1)
        {
            cout << "\nQueue overflow";
        }
        else
        {
            rear++;
            if (rear > 4)
                rear = 0;
        }
    }
};
```

```
        queue[rear] = x;
    }
}
void deleteQ()
{
    cout << "\nDeleted Element => " << queue[front];
    front++;
    if (front > 4)
        front = 0;
    cout << endl;
}
};

int main()
{
    Circular c;
    cout << endl
        << "Parth Vasoya \n 18DCS134" << endl;
    int choice, val;
    char y = 'n';
    do
    {
        cout << "1.Insert\n2.Delete\n";
        cin >> choice;
        switch (choice)
        {
            case 1:
                cout << "Enter value you want to insert :";
                cin >> val;
                c.insert(val);
                break;
            case 2:
                c.deleteQ();
                break;
        }
        cout << "Continue or not [y/n]";
        cin >> y;
    } while (y == 'y');
}
```

OUTPUT:

```

Parth Vasoya
18DCS134
1.Insert
2.Delete
1
Enter value you want to insert :10
Continue or not [y/n]y
1.Insert
2.Delete
1
Enter value you want to insert :50
Continue or not [y/n]y
1.Insert
2.Delete
1
Enter value you want to insert :40
Continue or not [y/n]y
1.Insert
2.Delete
1
Enter value you want to insert :80
Continue or not [y/n]y
1.Insert
2.Delete
2
Deleted Element => 10
Continue or not [y/n]y
1.Insert
2.Delete
1
Enter value you want to insert :200
Continue or not [y/n]y
1.Insert
2.Delete
1
Enter value you want to insert :70
Continue or not [y/n]y
1.Insert
2.Delete
1
Deleted Element => 10
Continue or not [y/n]y
1.Insert
2.Delete
1
Deleted Element => 40
Continue or not [y/n]y
1.Insert
2.Delete
2
Deleted Element => 80
Continue or not [y/n]n
Process returned 0 (0x0)   execution time
Press any key to continue.

```

CONCLUSION:

From this practical we have learnt about circular queue and implemented it using array.

PRACTICAL 8

AIM:

Implement the program display Linked List in Reverse.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
} * head;

void createList(int n);
void reverseList();
void displayList();

int main()
{
    int n, choice;
    printf("\nParth Vasoya \n 18DCS134\n");
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);
    printf("\nData in the list \n");

    displayList();
    printf("\nPress 1 to reverse the order of singly linked list\n");
    scanf("%d", &choice);
    if (choice == 1)
    {
        reverseList();
    }
    printf("\nData in the list\n");
    displayList();

    return 0;
```

```
}

void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;
    if (n <= 0)
    {
        printf("List size must be greater than zero.\n");
        return;
    }
    head = (struct node *)malloc(sizeof(struct node));
    if (head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);
        head->data = data;
        head->next = NULL;
        temp = head;
        for (i = 2; i <= n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));
            if (newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);
                newNode->data = data;
                newNode->next = NULL;
                temp->next = newNode;
                temp = temp->next;
            }
        }
    }
}
```

```
    }  
    printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");  
}  
}
```

```
void reverseList()  
{  
    struct node *prevNode, *curNode;  
    if (head != NULL)  
    {  
        prevNode = head;  
        curNode = head->next;  
        head = head->next;  
        prevNode->next = NULL;  
        while (head != NULL)  
        {  
            head = head->next;  
            curNode->next = prevNode;  
            prevNode = curNode;  
            curNode = head;  
        }  
        head = prevNode;  
        printf("SUCCESSFULLY REVERSED LIST\n");  
    }  
}
```

```
void displayList()  
{  
    struct node *temp;  
    if (head == NULL)  
    {  
        printf("List is empty.");  
    }  
    else  
    {  
        temp = head;  
        while (temp != NULL)  
        {  
            printf("Data = %d\n", temp->data);  
            temp = temp->next;  
        }  
    }  
}
```

```
}  
}  
}
```

OUTPUT:

```
Parth Vasoya  
18DCS134  
Enter the total number of nodes: 5  
Enter the data of node 1: 1  
Enter the data of node 2: 2  
Enter the data of node 3: 3  
Enter the data of node 4: 4  
Enter the data of node 5: 5  
SINGLY LINKED LIST CREATED SUCCESSFULLY  
  
Data in the list  
Data = 1  
Data = 2  
Data = 3  
Data = 4  
Data = 5  
  
Press 1 to reverse the order of singly linked list  
1  
SUCCESSFULLY REVERSED LIST  
  
Data in the list  
Data = 5  
Data = 4  
Data = 3  
Data = 2  
Data = 1  
  
Process returned 0 (0x0)   execution time : 11.071 s
```

CONCLUSION:

We have studied about how to access nodes in linked list to reverse side.

PRACTICAL 9

AIM:

Implement a city database using a BST to store the database records. Each database record contains the name of the city (a string of arbitrary length) and the coordinates of the city expressed as integer x- and y-coordinates. The BST should be organized by city name. Your database should allow records to be inserted, deleted by name or coordinate, and searched by name or coordinate. Another operation that should be supported is to print all records within a given distance of a specified point. Collect running-time statistics for each operation. Which operations can be implemented reasonably efficiently (i.e., in $\Theta(\log n)$ time in the average case) using a BST? Can the database system be made more efficient by using one or more additional BSTs to organize the records by location?

PROGRAM:

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
class BST
{
    string city;
    int x,y;
    BST *left,*right;
```

```
public:
    BST()
    {
        city="";
        x=0;
        y=0;
        left=right=NULL;
    }
```

```
    BST(string city,int x,int y)
    {
        this->city=city;
        this->x=x;
        this->y=y;
        left=right=NULL;
    }
```

```
    BST* Insert(BST *root,string city,int x,int y)
```

```
{
    if(!root)
    {
        return new BST(city,x,y);
    }

    if((city.at(0))>(root->city.at(0)))
    {
        root->right=Insert(root->right,city,x,y);
    }

    else
    {
        root->left=Insert(root->left,city,x,y);
    }

    return root;
}

void Inorder(BST *root)
{
    if(!root)
    {
        return;
    }

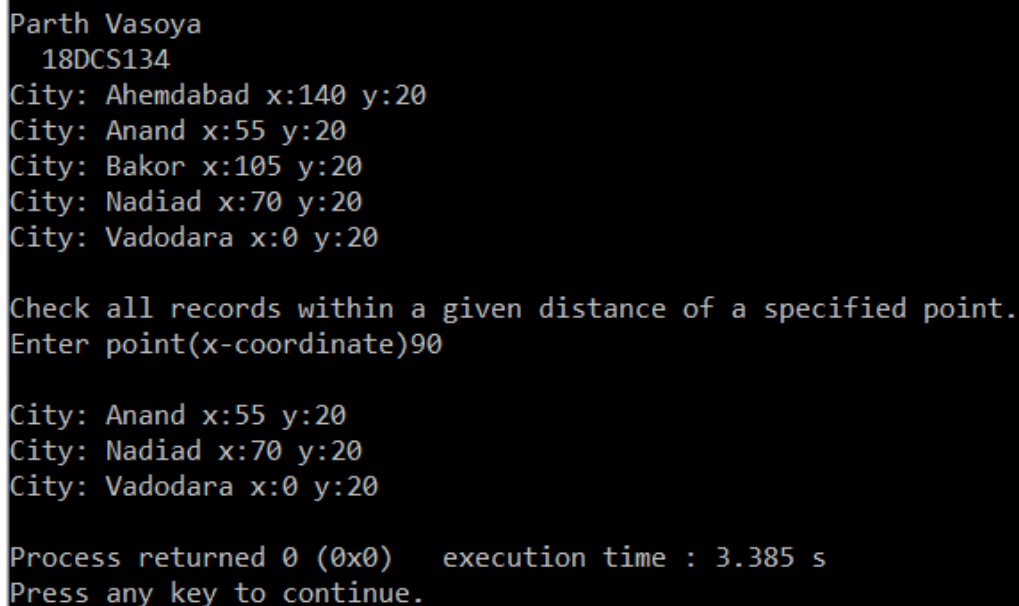
    Inorder(root->left);
    cout<<"City: "<<root->city<<" x:"<<root->x<<" y:"<<root->y<<"\n";
    Inorder(root->right);
}

void InorderByLocation(BST *root,int location)
{
    if(!root)
    {
        return;
    }

    InorderByLocation(root->left,location);
    if(location>=(root->x))
    {
        cout<<"City: "<<root->city<<" x:"<<root->x<<" y:"<<root->y<<"\n";
    }
    InorderByLocation(root->right,location);
}
};
```

```
int main()
{
    BST b,*root=NULL;
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    root=b.Insert(root,"Nadiad",70,20);
    b.Insert(root,"Vadodara",0,20);
    b.Insert(root,"Anand",55,20);
    b.Insert(root,"Ahemdabad",140,20);
    b.Insert(root,"Bakor",105,20);
    b.Inorder(root);
    cout<<"\nCheck all records within a given distance of a specified point.";
    cout<<"\nEnter point(x-coordinate)";
    int location;
    cin>>location;
    cout<<endl;
    b.InorderByLocation(root,location);

    return 0;
}
```

OUTPUT:A screenshot of a terminal window with a black background and white text. The output shows the program's execution: it prints the author's name and ID, then lists five cities with their x and y coordinates. It then prompts for a point to check, and lists the cities within a distance of 90. Finally, it shows the process returned 0 and the execution time.

```
Parth Vasoya
18DCS134
City: Ahemdabad x:140 y:20
City: Anand x:55 y:20
City: Bakor x:105 y:20
City: Nadiad x:70 y:20
City: Vadodara x:0 y:20

Check all records within a given distance of a specified point.
Enter point(x-coordinate)90

City: Anand x:55 y:20
City: Nadiad x:70 y:20
City: Vadodara x:0 y:20

Process returned 0 (0x0)   execution time : 3.385 s
Press any key to continue.
```

CONCLUSION:

From this practical we have learnt about circular queue and implemented it using array.

PRACTICAL 10

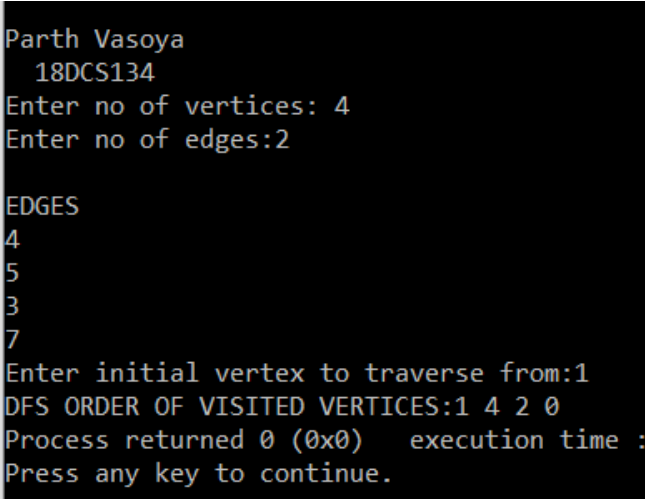
AIM:

Write a program that enters vertices, edges of a Graph and display sequence of vertices to traverse the graph in Depth First Search method.

PROGRAM:

```
#include<iostream>
using namespace std;
int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];
int main()
{
    int m;
    cout<<endl<<"Parth Vasoya \n 18DCS134"<<endl;
    cout<<"Enter no of vertices: ";
    cin>>n;
    cout <<"Enter no of edges:";
    cin >> m;
    cout <<"\nEDGES \n";
    for(k=1; k<=m; k++)
    {
        cin >>i>>j;
        cost[i][j]=1;
    }
    cout <<"Enter initial vertex to traverse from:";
    cin >>v;
    cout <<"DFS ORDER OF VISITED VERTICES:";
    cout << v <<" ";
    visited[v]=1;
    k=1;
    while(k<n)
    {
        for(j=n; j>=1; j--)
            if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
            {
                visit[j]=1;
                stk[top]=j;
                top++;
            }
        v=stk[--top];
        cout<<v <<" ";
        k++;
        visit[v]=0;
    }
```

```
        visited[v]=1;
    }
    return 0;
}
```

OUTPUT:

```
Parth Vasoya
18DCS134
Enter no of vertices: 4
Enter no of edges:2

EDGES
4
5
3
7
Enter initial vertex to traverse from:1
DFS ORDER OF VISITED VERTICES:1 4 2 0
Process returned 0 (0x0)   execution time :
Press any key to continue.
```

CONCLUSION:

In this practical, we learned about non linear data type Graph and Depth First Search and implemented it.

PRACTICAL 11

AIM:

In an array of 20 elements, arrange 15 different values, which are generated randomly. Use hash function to generate key and linear probing to avoid collision. $H(x) = (x \bmod 18) + 2$. Write a program to input and display the final values of array.

PROGRAM CODE:

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

main()
{
    int numbers[20];
    int array[20];
    int i, temp;
    int c, n;
    printf("Parth Vasoya \n 18dcs134\n");
    for (c = 0; c <= 14; c++)
    {
        n = rand() % 100 + 1;
        numbers[c] = n;
    }
    printf("\nThe random array is : ");
    for (c = 0; c <= 14; c++)
    {
        printf("\n element:%d is:- %d", c, numbers[c]);
    }
    for (i = 0; i < 18; i++)
    {
        temp = (numbers[i] % 18) + 2;
        array[temp] = numbers[i];
    }

    printf("\nThe key value pair are : ");
    for (i = 2; i < 20; i++)
    {
        printf("\n%d : \t%d", i, array[i]);
    }
}
```

OUTPUT:

```
Parth Vasoya
18dcs134

The random array is :
element:0 is:- 42
element:1 is:- 68
element:2 is:- 35
element:3 is:- 1
element:4 is:- 70
element:5 is:- 25
element:6 is:- 79
element:7 is:- 59
element:8 is:- 63
element:9 is:- 65
element:10 is:- 6
element:11 is:- 46
element:12 is:- 82
element:13 is:- 28
element:14 is:- 62
The key value pair are :
2 : 6818920
3 : 1
4 : 4199232
5 : 1988128749
6 : 1988128697
7 : 59
8 : 6
9 : 79
10 : 62
11 : 63
12 : 4200976
13 : 65
14 : 6356712
15 : 6818924
16 : 68
17 : 1988194293
18 : 70
19 : 35
Process returned 0 (0x0)   executi
Press any key to continue.
```

CONCLUSION:

In this practical, we learned about Hashing and Hash functions and implemented it using C++.