Department of IT and Computer Science

Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan

COMP-201L Data Structures and Algorithms Lab

Lab Report: 03

Class:	**Computer Science**
Name:	**Yaseen Ejaz Ahmed**
Registration No.:	**B20F0283CS014**
Semester:	**Third**
Submission Date:	
Submitted to:	**Dr. Rafi Ullah**

Instructor Signature

Lab No. 3

Recursion & Searching Algorithms

Objectives:

- To understand & implement the working of recursive functions in C++.
- To understand & Implement searching algorithms in C++.

Tools/Software Required:

C++ Compiler

Introduction:

A function that calls itself is known as recursive function. And, this technique is known as recursion. The figure below shows how recursion works by calling itself over and over again.

We can search for items in an array using different searching algorithms such as linear or binary search

Lab Tasks:

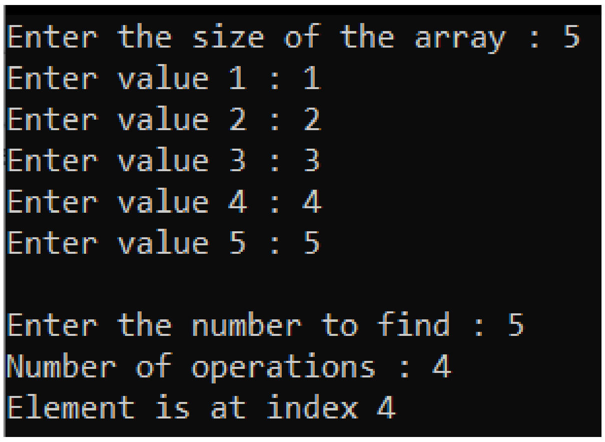
Lab Task 01: Write a program to find out a number among all other numbers entered by user using linear search technique.

Code:

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

int LinearSearch(int a[],int n,int num)
{
        int c;
        for(int i=0;i<n;i++)
        {
                if(a[i]==num)
                {
                        cout<<"Number of operations : "<<c;
                        return i;
                }
                c++;
        }
        return -1;


}

int main()
{
        int n,num;

        cout<<"Enter the size of the array : ";
        cin>>n;

        int a[n];

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

```
	cout<<"\nEnter the number to find : ";
	cin>>num;

	int index = LinearSearch(a,n,num);

	if(index == -1)
    cout<<"\nElement is not in the array";

    else
    cout<<"\nElement is at index "<<index;


}
```

Output:



```
Enter the size of the array : 5
Enter value 1 : 1
Enter value 2 : 2
Enter value 3 : 3
Enter value 4 : 4
Enter value 5 : 5

Enter the number to find : 5
Number of operations : 4
Element is at index 4
```

Lab Task 02: Write a program to find out a number among all other numbers entered by user using Binary search technique.

Code:

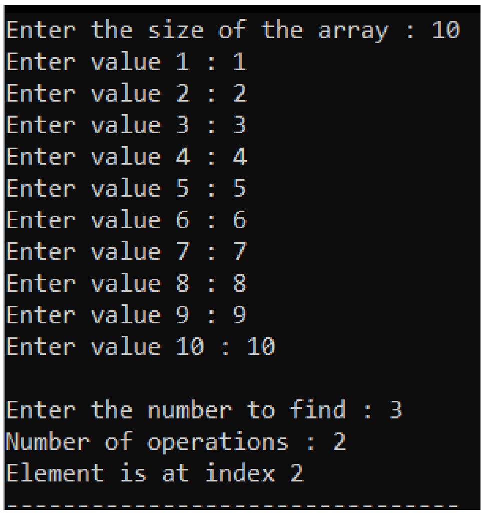
```
#include <iostream>
using namespace std;

int binarySearch(int a[],int n,int num)
{
	int start = 0,c=0;
	int end = n-1;
```

```
    while (start <= end)
    {
        int mid = (start+end)/2;

        if (a[mid] == num)
        {
            cout<<"Number of operations : "<<c;
            return mid;
         }


        else if (a[mid] < num)
        start = mid + 1;

        else if(a[mid] > num)
        end = mid - 1;

        c++;
    }
}
```

int main()

```
{
    int n,num;

        cout<<"Enter the size of the array : ";
        cin>>n;

        int a[n];

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

        cout<<"\nEnter the number to find : ";
        cin>>num;

    int index = binarySearch(a,n,num);
if(index == -1)
    cout<<"\nElement is not in the array";

    else
    cout<<"\nElement is at index "<<index;
}
```

Output:



```
Enter the size of the array : 10
Enter value 1 : 1
Enter value 2 : 2
Enter value 3 : 3
Enter value 4 : 4
Enter value 5 : 5
Enter value 6 : 6
Enter value 7 : 7
Enter value 8 : 8
Enter value 9 : 9
Enter value 10 : 10

Enter the number to find : 3
Number of operations : 2
Element is at index 2
--------------------------------
```

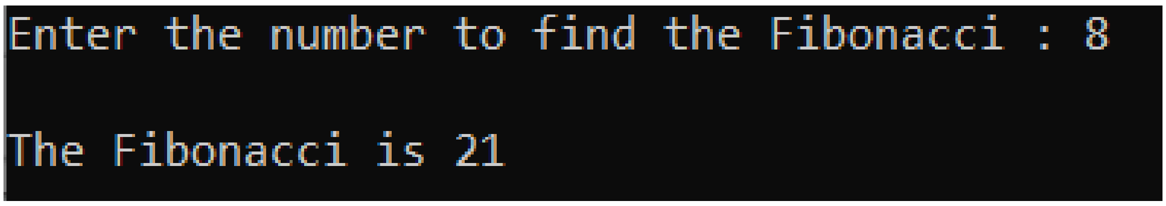
Lab Task 03: Find Sum of Fibonacci Series using Recursive Function

Code:

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


int fibo(int n)
{
        if(n<=0)
        return 0;

        else if(n==1)
        return 1;

        else
        return fibo(n-1) + fibo(n-2);
}
```

```
int main()
{
	int n;

	cout<<"Enter the number to find the Fibonacci : ";
	cin>>n;

	cout<<"\nThe Fibonacci is "<<fibo(n);
}
```

Output:



```
Enter the number to find the Fibonacci : 8

The Fibonacci is 21
```

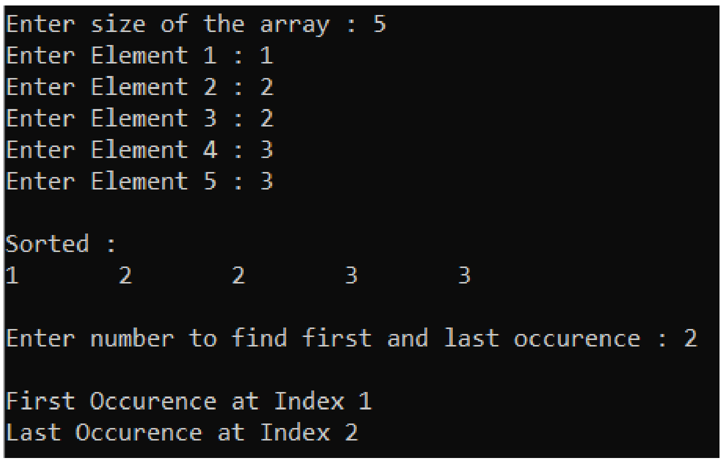
In Lab Task 01: Given a sorted array of integers, find index of first or last occurrence of a given number. If the element is not found in the array, report that as well.

Code:

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

Sorted(int a[],int size)
{
	int flag=0;
	int temp,n;
	for(int i=0;i<size;i++)
	{
		for(int j=i+1;j<size;j++)
		{
			if(a[i]>a[j])
			{
				temp=a[i];
				a[i]=a[j];
				a[j]= temp;
			}
		}
	}
	cout<<"\nSorted : \n";
	for(int i=0;i<size;i++)
	{
		cout<<a[i]<<"\t";
	}

	cout<<"\n\nEnter number to find first and last occurence : ";
	cin>>n;


	for(int i=0;i<size;i++)
	{
		if(a[i]==n)
		{
			cout<<"\nFirst Occurence at Index "<<i;
			flag++;
			break;
		}
	}

	for(int i=size-1;i>=0;i--)
	{
		if(a[i]==n)
		{
			cout<<"\nLast Occurence at Index "<<i;
			flag++;
			break;
		}
	}

	if(flag==0)
	cout<<"\nElement Not Found in the Array";
}

int main()
{
	int size;
	cout<<"Enter size of the array : ";
	cin>>size;

	int a[size];
```

```
        for(int i=0;i<size;i++)
        {
                cout<<"Enter Element "<<i+1<<" : ";
                cin>>a[i];
        }

        Sorted(a,size);
}
```

Output:



```
Enter size of the array : 5
Enter Element 1 : 1
Enter Element 2 : 2
Enter Element 3 : 2
Enter Element 4 : 3
Enter Element 5 : 3

Sorted :
1       2       2       3       3

Enter number to find first and last occurence : 2

First Occurence at Index 1
Last Occurence at Index 2
```

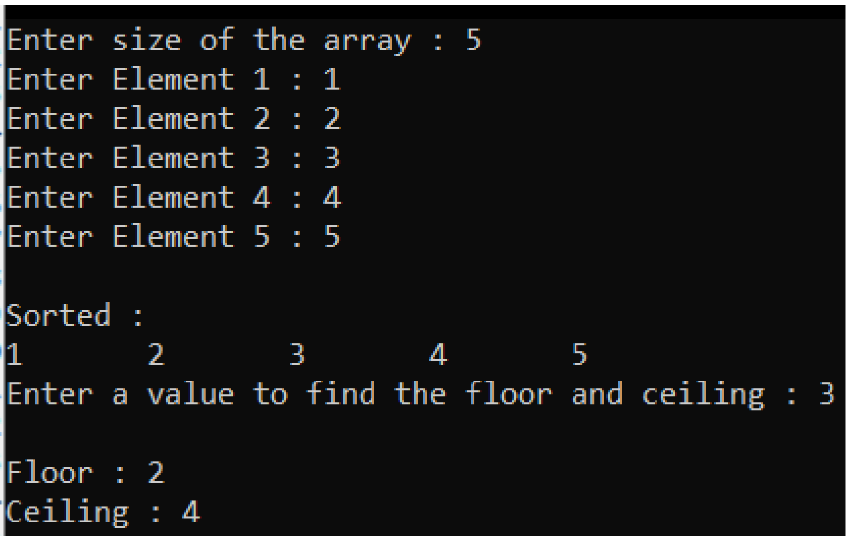
In Lab Task 02: Given a sorted array of integers, find floor and ceil of a given number in it. The floor and ceil map the given number to the largest previous or the smallest following integer, respectively.

Code:

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

Sorted(int a[],int size)
{
        int temp,n,floor=0,ceiling=0;
        for(int i=0;i<size;i++)
        {
                for(int j=i+1;j<size;j++)
                {
                        if(a[i]>a[j])
                        {
                                temp=a[i];
                                a[i]=a[j];
                    a[j]= temp;
                        }
                }
        }

        cout<<"\nSorted : \n";
        for(int i=0;i<size;i++)
        {
                cout<<a[i]<<"\t";
        }

        cout<<"\nEnter a value to find the floor and ceiling : ";
        cin>>n;

        for(int i=size-1;i>=0;i--)
        {
                if(n>a[i])
                {
                        cout<<"\nFloor : "<<a[i];
                        floor++;
                        break;
                }
        }

        if(floor==0)
        cout<<"\nFLOOR DOES NOT EXIST";

        for(int i=0;i<size;i++)
        {
                if(n<a[i])
                {
                        cout<<"\nCeiling : "<<a[i];
                        ceiling++;
                        break;
                }
        }

        if(ceiling==0)
        cout<<"\nFLOOR DOES NOT EXIST";


}
```

```
int main()
{
        int size;
        cout<<"Enter size of the array : ";
        cin>>size;

        int a[size];

        for(int i=0;i<size;i++)
        {
                cout<<"Enter Element "<<i+1<<" : ";
                cin>>a[i];
        }

        Sorted(a,size);
}
```

Output:



```
Enter size of the array : 5
Enter Element 1 : 1
Enter Element 2 : 2
Enter Element 3 : 3
Enter Element 4 : 4
Enter Element 5 : 5

Sorted :
1       2       3       4       5
Enter a value to find the floor and ceiling : 3

Floor : 2
Ceiling : 4
```

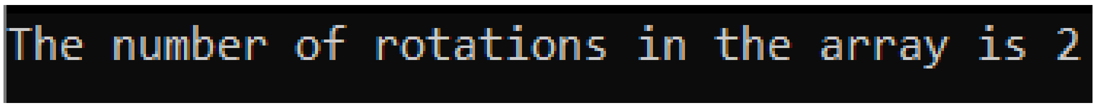
In Lab Task 03: Given a circularly sorted array of integers, find the number of times the array is rotated. Assume there are no duplicates in the array and the rotation is in anti-clockwise direction.
Input : arr = [9, 10, 2, 5, 6, 8]
Output: The array is rotated 2 times

Code:

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

int main()
```

```
{

        int a[6] = {2,5,6,8,9,10};
        int c=0;
        int b[6] = {9, 10, 2, 5, 6, 8};
        for(int i=0;i<6;i++)
        {
                if(a[0]!=b[i])
                c++;

                else
                break;
        }

        cout<<"The number of rotations in the array is "<<c;

        delete[] a;
        delete[] b;
}
```

Output:



```
The number of rotations in the array is 2
```

----------OR----------

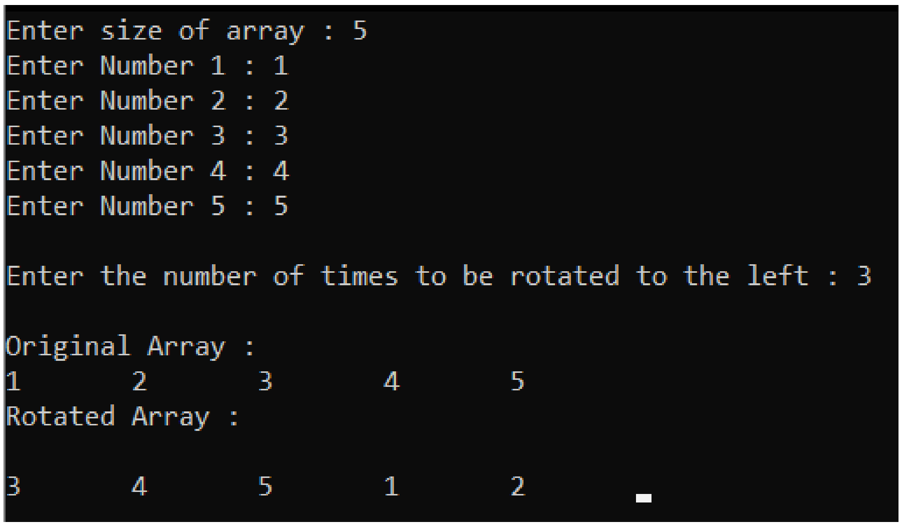
Code:

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

void RotateArray(int a[],int size,int rotate)
{
        int b[size];
        cout<<"\nOriginal Array : \n";
        for(int i=0;i<size;i++)
        {
                cout<<a[i]<<"\t";
        }

        cout<<"\nRotated Array : \n";
        for(int i=0;i<size;i++)
        {
                if(rotate>=size)        //outside range of size of array
           {
                rotate=0;          //move to index 0 the beginning in that case
                }

           b[rotate]=a[i];
           rotate++;
        }

        cout<<endl;
        for(int i=0;i<size;i++)
        cout<<b[i]<<"\t";
}
```

int main()

```
{
        int size,rotate;

        cout<<"Enter size of array : ";
        cin>>size;

        int a[size];
        int b[size];

        for(int i=0;i<size;i++)
        {
                cout<<"Enter Number "<<i+1<<" : ";
                cin>>a[i];
        }

        cout<<"\nEnter the number of times to be rotated to the left : ";
        cin>>rotate;

        RotateArray(a,size,rotate);

        delete[] a;
        delete[] b;
}
```

Output:



```
Enter size of array : 5
Enter Number 1 : 1
Enter Number 2 : 2
Enter Number 3 : 3
Enter Number 4 : 4
Enter Number 5 : 5

Enter the number of times to be rotated to the left : 3

Original Array :
1       2       3       4       5
Rotated Array :

3       4       5       1       2
```

Results & Observations:

In this lab, we have learnt the basics of recursion and searching algorithms. We can use recursion and different types of searches for different problems depending on the inputs and outputs.