Course Name: A Level (2nd Sem) **Subject:** Data Structure using C++
Topic: Binary Search in C++ **Date:** 22-04-2020

Example1 : Binary Search

A binary search (or half-interval search) is applicable only to a sorted array. It compares the search key with the middle element. If there is a match, it returns the element's index. If the search key is less then the middle element, repeat searching on the left half; otherwise, search the right half. If the remaining element to be searched is zero, return "no found".

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

int binarySearch(const int a[], int size, int key);
int binarySearch(const int a[], int iLeft, int iRight, int key);
void print(const int a[], int iLeft, int iRight);

int main()
{
   const int SIZE = 10;
   int a1[SIZE] = {1, 4, 5, 8, 12, 19, 24, 31, 43, 55}; // sorted

   cout << binarySearch(a1, SIZE, 8) << endl;
   cout << binarySearch(a1, SIZE, 12) << endl;
   cout << binarySearch(a1, SIZE, 24) << endl;
   cout << binarySearch(a1, SIZE, 21) << endl;
}

// Search the array for the given key
// If found, return array index; otherwise, return -1
int binarySearch(const int a[], int size, int key)
{
   // Call recursive helper function
   return binarySearch(a, 0, size-1, key);
}

// Recursive helper function for binarySearch
int binarySearch(const int a[], int iLeft, int iRight, int key)
{
   // For tracing the algorithm
   print(a, iLeft, iRight);

   // Test for empty list
   if (iLeft > iRight) return -1;

   // Compare with middle element
   int mid = (iRight + iLeft) / 2;  // truncate
   if (key == a[mid])
  {
      return mid;
   } else if (key < a[mid])
   {
```

```
    // Recursively search the lower half
    binarySearch(a, iLeft, mid - 1, key);
  }
else
  {
    // Recursively search the upper half
    binarySearch(a, mid + 1, iRight, key);
  }
}

// Print the contents of the given array from iLeft to iRight (inclusive)
void print(const int a[], int iLeft, int iRight)
{
  cout << "{";
  for (int i = iLeft; i <= iRight; ++i)
 {
    cout << a[i];
    if (i < iRight) cout << ",";
  }
  cout << "}" << endl;
}
```

Complexity

The worst-case and average-case time complexity for binary search is O(log n). The best-case is O(1).