Binary Search

Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one.

Binary Search Pseudocode:

Iteration Method:

```
procedure binarySearch(arr, x, left, right)
       while repeat till left = right
              mid = (left+right)/2
              if (x == arr[mid])
                     return mid
              end if
              else if
                     left = mid + 1
              end else if
              else
                     right = mid - 1
              end else
       end while
end procedure
```

Recursive Method:

```
procedure binarySearch(arr, x, left, right)
       if left > right
              return False
       end if
       else
              mid = (left + right) / 2
              if x == arr[mid]
                     return mid
              end if
              else if x > arr[mid]
                     return binarySearch(arr, x, mid + 1, right)
              end else if
              else
                     return binarySearch(arr, x, right, mid - 1)
              end else
       end else
end procedure
```

Complexities: Time Complexity: Best – O(1), Average – O(logn), Worst – O(logn)
Space Complexity: O(1)

Applications: In libraries of Java, .Net, C++ STL While debugging, the binary search is used to pinpoint the place where the error happens.

Source Code: Iterative Method

```
using System;
namespace BinarySearch
{
    class Program
    {
        public static void Main(String[] args)
        {
            Input();
        }
        public static void Input()
        {
            Console.Write("Enter the number of items: ");
            int numberOfItems = Convert.ToInt32(Console.ReadLine());
            int[] itemsList = new int[numberOfItems];
            Console.WriteLine("Enter the items: ");
            for (int i = 0; i < itemsList.Length; i++)
            {
                itemsList[i] = Convert.ToInt32(Console.ReadLine());
            }

            Console.Write("Enter the searching item: ");
            int searchItem = Convert.ToInt32(Console.ReadLine());
            int result = BinarySearch(itemsList, searchItem);
            if(result == -1)
            {
                Console.WriteLine("Item does not find");
            }
            else
            {
                Console.WriteLine($"Item is found in {result+1} position");
            }
        }
        public static int BinarySearch(int[] itemsList, int searchItem)
        {
            int left = 0;
            int right = itemsList.Length - 1;
            while (left <= right)
            {
                int mid = (left + right) / 2;
                if (itemsList[mid] == searchItem)
                {
                    return mid;
                }
                else if(itemsList[mid] < searchItem)
                {
                    left = mid + 1;
                }
                else
                {
                    right = mid - 1;
                }
            }
            return -1;
        }
    }
}
```

Source Code: Recursive Method

```
using System;

namespace RecursiveBinarySearch
{
    class Program
    {
        public static void Main(String[] args)
        {
            Input();
        }
        public static void Input()
        {
            Console.Write("Enter the number of items: ");
            int numberOfItems = Convert.ToInt32(Console.ReadLine());
            int[] itemsList = new int[numberOfItems];
            Console.WriteLine("Enter the items: ");
            for (int i = 0; i < itemsList.Length; i++)
            {
                itemsList[i] = Convert.ToInt32(Console.ReadLine());
            }
            Console.Write("Enter the searching item: ");
            int searchItem = Convert.ToInt32(Console.ReadLine());

            int result= RBinarySearch(itemsList, searchItem, 0, itemsList.Length-1);

            if (result == -1)
            {
                Console.WriteLine("Item does not find");
            }
            else
            {
                Console.WriteLine($"Item is found in {result + 1} position");
            }
        }
        static int RBinarySearch(int[] itemsList,int searchItem,int left,int right)
        {
            if(left <= right)
            {
                int mid = (right - left) / 2;
                if(itemsList[mid] == searchItem)
                {
                    return mid;
                }
                if(itemsList[mid] > searchItem)
                {
                    return RBinarySearch(itemsList, searchItem, left, mid-1);
                }
                else
                {
                    return RBinarySearch(itemsList, searchItem, mid+1, right);
                }
            }
            return -1;
        }
    }
}
```