**Practical No : 1**

**Name :** Samrudhi Pawar

**Roll No :** 44

**Title :** Develop a program to design a function for Binary Search using Divide and Conquer Strategies. Also compute it's time complexity.

**Program :**

#include <iostream> // Includes the iostream library, which allows input and output (cin, cout).

#include <vector> // Includes the vector library, which provides dynamic array functionality.

#include <algorithm> // for sorting the array

using namespace std; / Allows us to use standard functions (like cout and cin) without the "std::" prefix.

// Binary Search function

int binSearch(const vector<int>& arr, int low, int high, int key) {

    if (high < low) // Base case: element not found

        return -1;

    int mid = (low + high) / 2; // Find the middle index

    if (arr[mid] == key)        // If element is found at mid

        return mid;

    else if (arr[mid] > key)    // If element is smaller than mid

        return binSearch(arr, low, mid - 1, key); // Search left half

    else                        // If element is larger than mid

        return binSearch(arr, mid + 1, high, key); // Search right half

}

//This function is recursive, meaning it calls itself to divide the problem into smaller sub-problems.

int main() {

    int n, key;

    // Input size of the array

    cout << "Enter the size of array: ";

    cin >> n;

    vector<int> arr(n); // Declare a vector to hold the array elements

    // Input the array elements

    for (int i = 0; i < n; i++) {

        cout << "Enter the value: ";

        cin >> arr[i];

    }

    // Sort the array before performing binary search

    sort(arr.begin(), arr.end());

    cout << "Array elements are: ";

    for (int i = 0; i < n; i++)

        cout << arr[i] << " ";

    cout << endl;

    // Input the key to search

    cout << "Enter the value to be searched: ";

    cin >> key;

    // Perform binary search

    int index = binSearch(arr, 0, n - 1, key);

    // Output the result

    if (index == -1)

        cout << "Element is not present......." << endl;

    else

        cout << "Element " << key << " is present at location: " << index + 1 << endl;

    return 0;

}

Output:

Enter the size of array: 4

Enter the value: 2

Enter the value: 6

Enter the value: 1

Enter the value: 0

Array elements are: 0 1 2 6

Enter the value to be searched: 2

Element 2 is present at location: 3 //best case:o(1),avg-worst case:o(log n)