The University of Azad Jammu and Kashmir

Submitted by: M.Amir

Subject: Discrete Structure & Algorithm

Roll no: 23

Submitted to: Sir Zeshan

Task: Linear Search & Binary Search, 04

Question 01:

Binary search is a fast way to find something in a sorted list.

Instead of checking every element one by one, it:

1. Looks at the middle of the list.
2. Decides whether the target is on the left side or the right side.
3. Cuts the list in half and repeats.

Because it keeps cutting the list in half, it is much faster than linear search.

⭑ Algorithm

Assume the array is sorted.

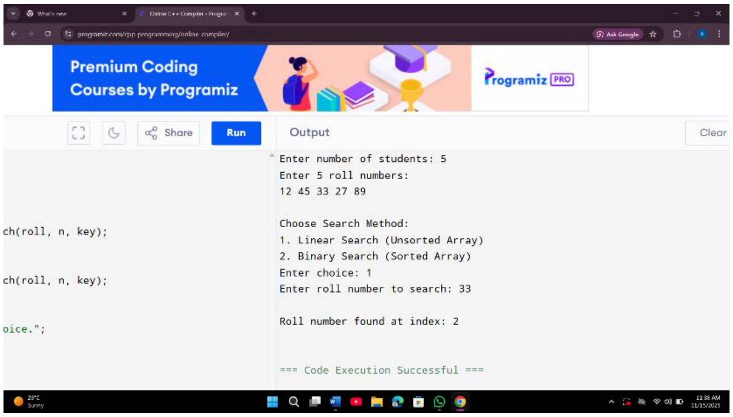
```
1. Set low = 0
2. Set high = n - 1
3. While low <= high
    a. Find mid = (low + high) / 2
    b. If array[mid] == key, then found
    c. If key < array[mid], search left half → high = mid - 1
    d. Else, search right half → low = mid + 1
4. If the loop ends, the key is not found.
```

Question 2:

```
#include <iostream>
using namespace std;
int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key)
            return i;
    }
    return -1;
}
int binarySearch(int arr[], int n, int key) {
    int left = 0, right = n - 1;
  while (left <= right) {
        int mid = (left + right) / 2;
  if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}
int main() {
    int n, choice, key;
    cout << "Enter number of students: ";
    cin >> n;
    int roll[n];
   cout << "Enter " << n << " roll numbers:\n";
    for (int i = 0; i < n; i++) {
        cin >> roll[i];
    }
  cout << "\nChoose Search Method:\n";
    cout << "1. Linear Search (Unsorted Array)\n";
    cout << "2. Binary Search (Sorted Array)\n";
    cout << "Enter choice: ";
    cin >> choice;
    cout << "Enter roll number to search: ";
    cin >> key;
 int result = -1;
    if (choice == 1) {
        result = linearSearch(roll, n, key);
    }
    else if (choice == 2) {
        result = binarySearch(roll, n, key);
    }
    else {
        cout << "Invalid choice.";
        return 0;
    }
    if (result != -1)
        cout << "\nRoll number found at index: " << result << endl;
    else
        cout << "\nRoll number NOT found in the list.\n";
   return 0;
}
```

Output:

Case 1:



```
ch(roll, n, key);

ch(roll, n, key);

oice.";
```

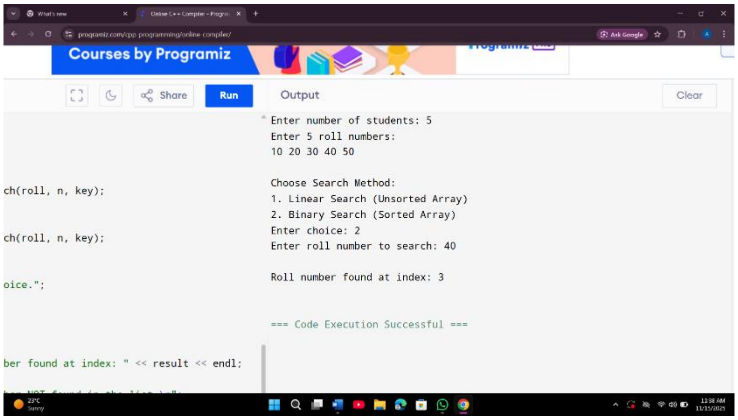
```
Enter number of students: 5
Enter 5 roll numbers:
12 45 33 27 89

Choose Search Method:
1. Linear Search (Unsorted Array)
2. Binary Search (Sorted Array)
Enter choice: 1
Enter roll number to search: 33

Roll number found at index: 2


=== Code Execution Successful ===
```

Case 2:



```
ch(roll, n, key);

ch(roll, n, key);

oice.";

ber found at index: " << result << endl;
```

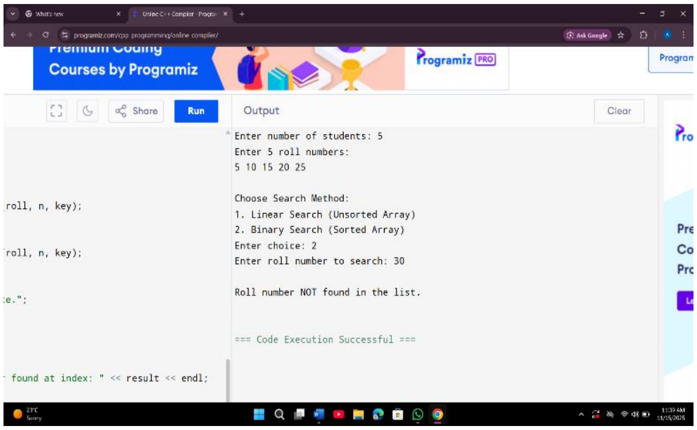
```
Enter number of students: 5
Enter 5 roll numbers:
10 20 30 40 50

Choose Search Method:
1. Linear Search (Unsorted Array)
2. Binary Search (Sorted Array)
Enter choice: 2
Enter roll number to search: 40

Roll number found at index: 3


=== Code Execution Successful ===
```

Case 3: Not found case



```
roll, n, key);

roll, n, key);

e.";

 found at index: " << result << endl;
```

```
Enter number of students: 5
Enter 5 roll numbers:
5 10 15 20 25

Choose Search Method:
1. Linear Search (Unsorted Array)
2. Binary Search (Sorted Array)
Enter choice: 2
Enter roll number to search: 30

Roll number NOT found in the list.


=== Code Execution Successful ===
```

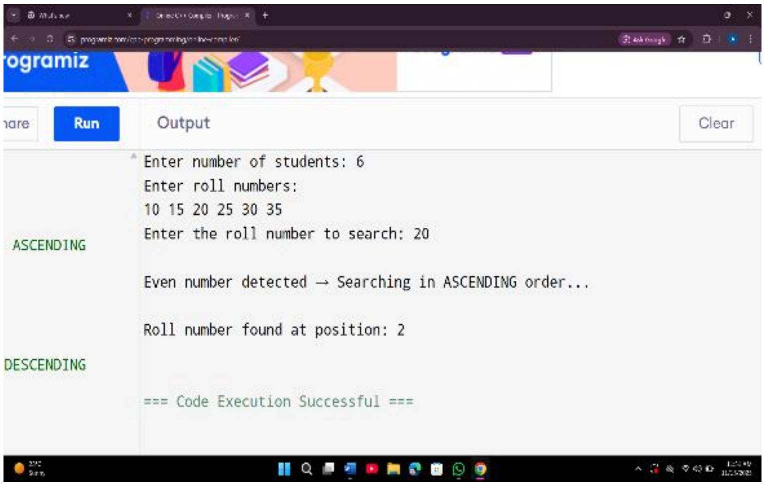
Question 3:

```
#include <iostream>
using namespace std;
int binarySearchAsc(int arr[], int n, int key) {

    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
int binarySearchDesc(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key)
            return mid;
        else if (arr[mid] > key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}
int main() {
    int n;
    cout << "Enter number of students: ";
    cin >> n;
    int arr[n];
    cout << "Enter roll numbers:\n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    int key;
    cout << "Enter the roll number to search: ";
    cin >> key;
    int position;
    if (key % 2 == 0) {
        cout << "\nEven number detected → Searching in ASCENDING order...\n";
        position = binarySearchAsc(arr, n, key);
    }
    else {
        cout << "\nOdd number detected → Searching in DESCENDING order...\n";
        position = binarySearchDesc(arr, n, key);
    }
    if (position != -1)
        cout << "\nRoll number found at position: " << position << endl;
    else
        cout << "\nRoll number not found in the array.\n";
    return 0;
}
```

Output:

Case 1:



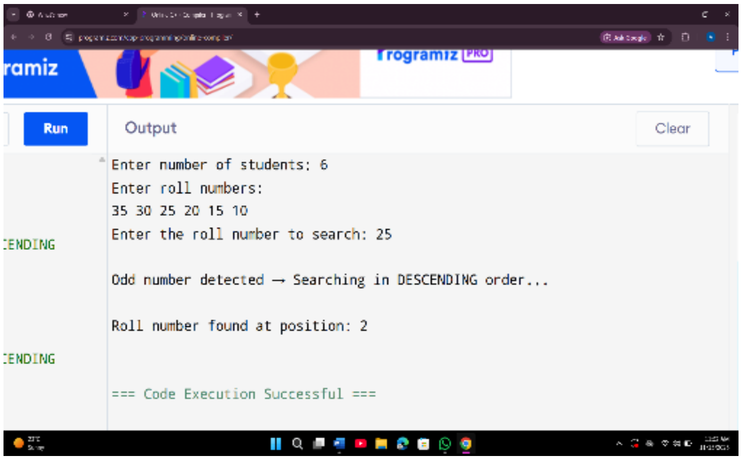
```
Enter number of students: 6
Enter roll numbers:
10 15 20 25 30 35
Enter the roll number to search: 20

Even number detected → Searching in ASCENDING order...

Roll number found at position: 2


=== Code Execution Successful ===
```

Case 2:



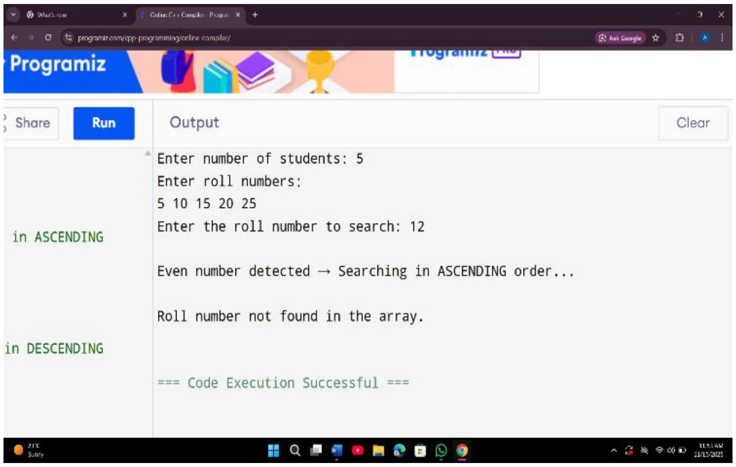
```
Enter number of students: 6
Enter roll numbers:
35 30 25 20 15 10
Enter the roll number to search: 25

Odd number detected → Searching in DESCENDING order...

Roll number found at position: 2


=== Code Execution Successful ===
```

Case 3:



```
 in ASCENDING

in DESCENDING
```

```
Enter number of students: 5
Enter roll numbers:
5 10 15 20 25
Enter the roll number to search: 12

Even number detected → Searching in ASCENDING order...

Roll number not found in the array.


=== Code Execution Successful ===
```