Day - 5

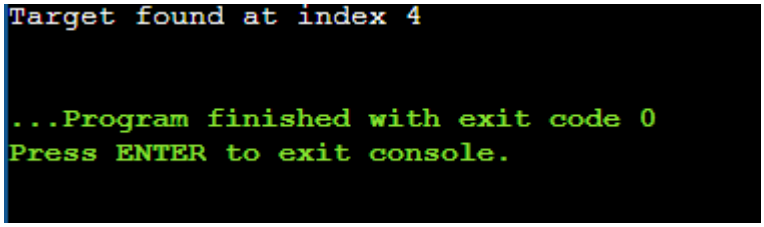
NAME- Lakshay Verma
UID-22BCS15481
DATE-26/12/2024

Question 1

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

int linearSearch(const vector<int>& arr, int target) {

    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] == target) {
            return i;
        }
    }
    return -1;
}

int main() {
    vector<int> array = {10, 20, 30, 40, 50};
    int targetValue = 50;

    int result = linearSearch(array, targetValue);

    if (result != -1) {
```

```
        cout << "Target found at index " << result << endl;
    } else {
        cout << "Target not found in the array" << endl;
    }

    return 0;
}
```

Output :

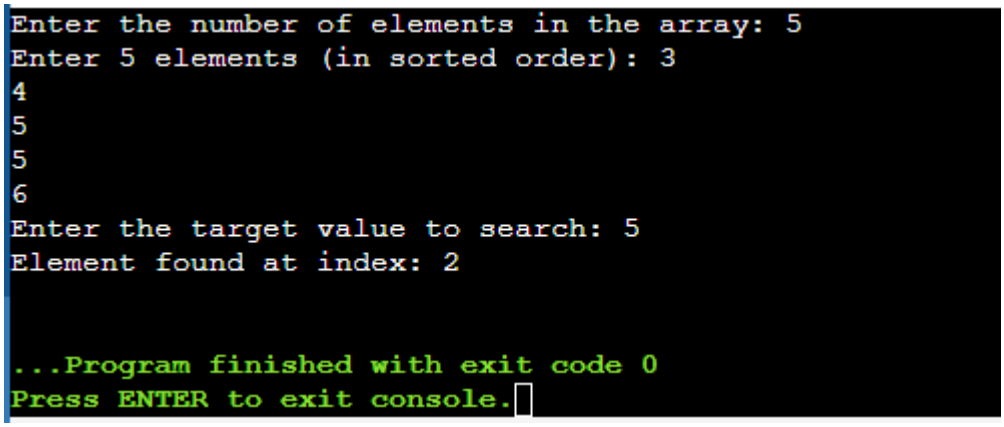


Question 2

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

int binarySearch(int arr[], int size, int target) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) {
            return mid;
        }
        if (arr[mid] < target) {
            left = mid + 1;
        }
        else {
            right = mid - 1;
```

```
        }
    }

    return -1;
}
int main() {
    int size, target;
    cout << "Enter the number of elements in the array: ";
    cin >> size;
    int arr[size];
    cout << "Enter " << size << " elements (in sorted order): ";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }
    cout << "Enter the target value to search: ";
    cin >> target;
    sort(arr, arr + size);
    int result = binarySearch(arr, size, target);
    if (result != -1) {
        cout << "Element found at index: " << result << endl;
    } else {
        cout << "Element not found" << endl;
    }

    return 0;
}
```

Output:

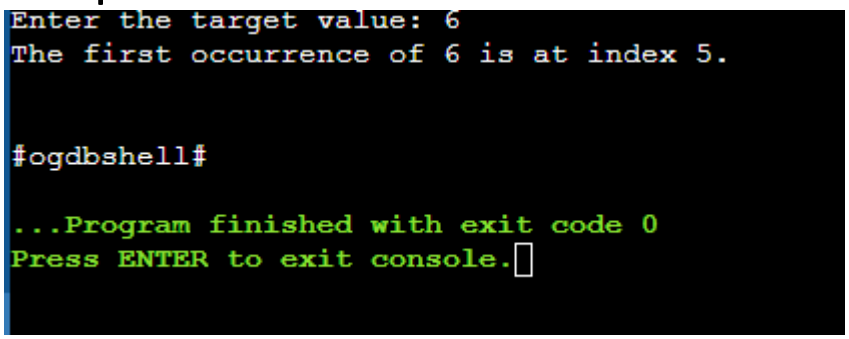


Question 3

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


int findFirstOccurrence(const vector<int>& arr, int target) {
    int left = 0, right = arr.size() - 1;
    int result = -1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            result = mid;
            right = mid - 1;
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
```

```
        }
    }

    return result;
}

int main() {
    vector<int> arr = {1, 2, 4, 4, 4, 6, 7};
    int target;

    cout << "Enter the target value: ";
    cin >> target;

    int index = findFirstOccurrence(arr, target);

    if (index != -1) {
        cout << "The first occurrence of " << target << " is at index
" << index << "." << endl;
    } else {
        cout << target << " is not present in the array." << endl;
    }

    return 0;
}
```

Output:

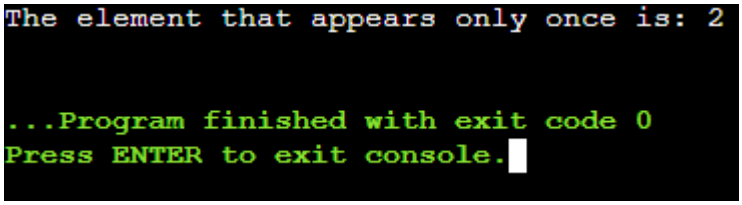


Question 4

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


int findSingleElement(const vector<int>& arr) {
    int left = 0, right = arr.size() - 1;

    while (left < right) {
        int mid = left + (right - left) / 2;


        if (mid % 2 == 1) {
            mid--;
        }


        if (arr[mid] == arr[mid + 1]) {

            left = mid + 2;
        } else {

            right = mid;
        }
    }


    return arr[left];
}

int main() {
    vector<int> arr = {1, 1, 2, 3, 3, 4, 4, 8, 8};

    int result = findSingleElement(arr);
```

```
    cout << "The element that appears only once is: " << result <<
endl;

    return 0;
}
```

Output:



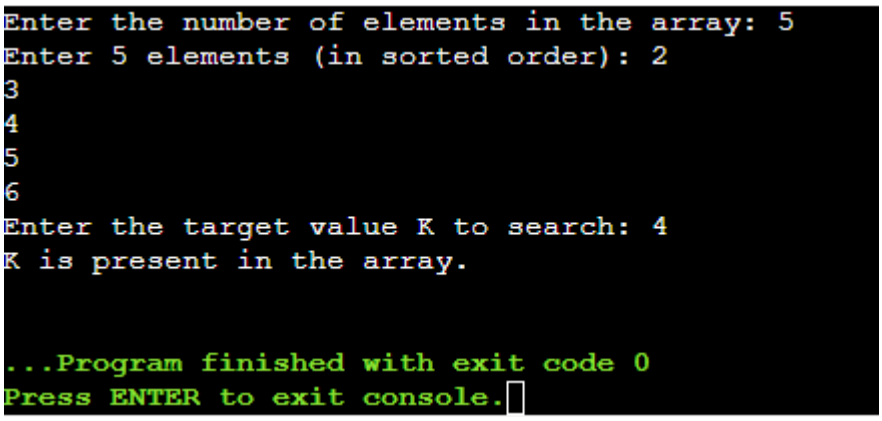
Question 5

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


bool binarySearch(int arr[], int size, int K) {
    int left = 0;
    int right = size - 1;


    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == K) {
            return true;
        }
        else if (arr[mid] < K) {
            left = mid + 1;
        }
```

```
        else {
            right = mid - 1;
        }
    }

}

int main() {
    int size, K;


    cout << "Enter the number of elements in the array: ";
    cin >> size;


    int arr[size];


    cout << "Enter " << size << " elements (in sorted order): ";
    for (int i = 0; i < size; i++) {
        cin >> arr[i];
    }


    cout << "Enter the target value K to search: ";
    cin >> K;


    if (binarySearch(arr, size, K)) {
        cout << "K is present in the array." << endl;
    } else {
        cout << "K is not present in the array." << endl;
    }

    return 0;
```

}

Output:

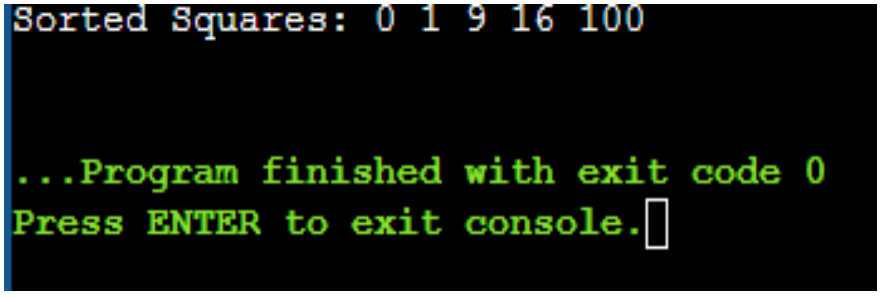


Question 6

```
#include <iostream>
#include <vector>
#include <algorithm>

std::vector<int> sortedSquares(const std::vector<int>& nums) {
    int n = nums.size();
    std::vector<int> result(n);
    int left = 0, right = n - 1;
    int pos = n - 1;

    while (left <= right) {
        int leftSquare = nums[left] * nums[left];
        int rightSquare = nums[right] * nums[right];
        if (leftSquare > rightSquare) {
            result[pos] = leftSqua

            result[pos] = rightSquare;
            right--;
```

```
        }
        pos--;
    }

    return result;
}

int main() {
    std::vector<int> nums = {-4, -1, 0, 3, 10};
    std::vector<int> result = sortedSquares(nums);

    std::cout << "Sorted Squares: ";
    for (int num : result) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

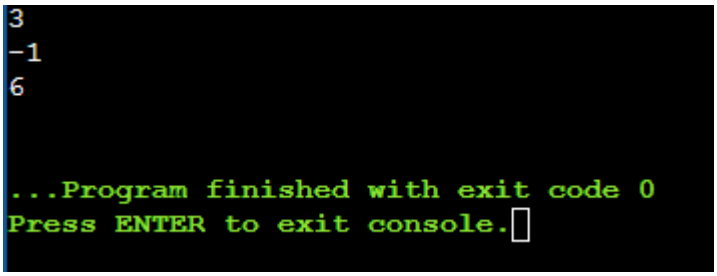
Output:



Question 7

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

int findFirstOccurrence(int k, const vector<int>& arr) {

   for (int i = 0; i < arr.size(); i++) {
      if (arr[i] == k) {
         return i + 1;
      }
   }
   return -1;
}

int main() {

   int k1 = 16;
   vector<int> arr1 = {9, 7, 16, 16, 4};
   cout << findFirstOccurrence(k1, arr1) << endl;


   int k2 = 98;
   vector<int> arr2 = {1, 22, 57, 47, 34, 18, 66};
   cout << findFirstOccurrence(k2, arr2) << endl;


   int k3 = 9;
   vector<int> arr3 = {1, 22, 57, 47, 34, 9, 66};
   cout << findFirstOccurrence(k3, arr3) << endl;

   return 0;
}
```

Output:

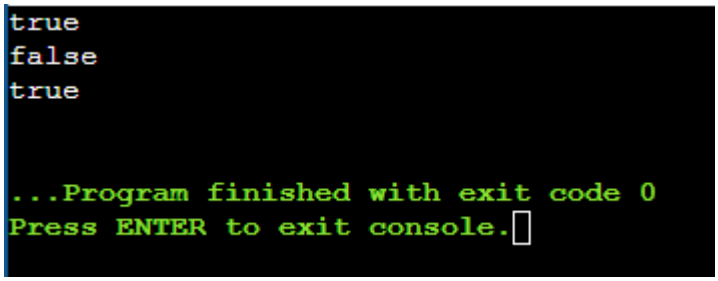


Question 8

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

bool isPresent(const vector<int>& arr, int k) {
    int left = 0, right = arr.size() - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == k) {
            return true;
        } else if (arr[mid] < k) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return false;
}

int main() {
    vector<int> arr1 = {1, 2, 3, 4, 6};
    int k1 = 6;
```

```
    cout << (isPresent(arr1, k1) ? "true" : "false") << endl;

    vector<int> arr2 = {1, 2, 4, 5, 6};
    int k2 = 3;
    cout << (isPresent(arr2, k2) ? "true" : "false") << endl;

    vector<int> arr3 = {1, 2, 4, 5, 6};
    int k3 = 6;
    cout << (isPresent(arr3, k3) ? "true" : "false") << endl;

    return 0;
}
```

Output:

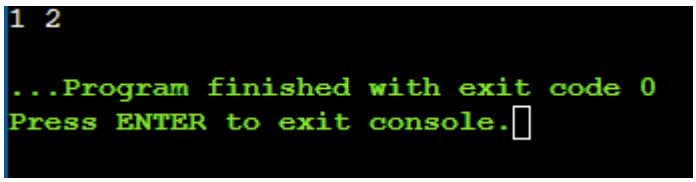


Question 9

```
#include <vector>
#include <iostream>

std::vector<int> targetIndices(const std::vector<int>& nums, int
target) {
    int lessCount = 0, targetCount = 0;


    for (int num : nums) {
        if (num < target) {
            ++lessCount;
```

```
        } else if (num == target) {
            ++targetCount;
        }
    }


    std::vector<int> result;
    for (int i = 0; i < targetCount; ++i) {
        result.push_back(lessCount + i);
    }

    return result;
}

int main() {
    std::vector<int> nums = {1, 2, 5, 2, 3};
    int target = 2;

    std::vector<int> result = targetIndices(nums, target);


    for (int index : result) {
        std::cout << index << " ";
    }

    return 0;
}
```

Output:



Question 10

```
#include <vector>
#include <iostream>

int searchInsert(std::vector<int>& nums, int target) {
    int left = 0, right = nums.size() - 1;


    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (nums[mid] == target) {
            return mid;
        } else if (nums[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }


    return left;
}

int main() {
    std::vector<int> nums1 = {1, 3, 5, 6};
    int target1 = 5;
    std::cout << "Output: " << searchInsert(nums1, target1) <<
"\n";

    std::vector<int> nums2 = {1, 3, 5, 6};
    int target2 = 2;
```

```
    std::cout << "Output: " << searchInsert(nums2, target2) <<
"\n";

    std::vector<int> nums3 = {1, 3, 5, 6};
    int target3 = 7;
    std::cout << "Output: " << searchInsert(nums3, target3) <<
"\n";

    return 0;
}
```

Output: