

Name: Nandita Dutta

UID: 22BCS15313

Section: 22BCS_IOT_620-A

Date: 26-12-24

DOMAIN WINTER WINNING CAMP-Day(5)

1) Search No.

Code:

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

int searchNumber(vector<int>& arr, int k) {
    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] == k) return i + 1; // 1-based indexing
    }
    return -1;
}

int main() {
    int n, k;
    cout << "Enter the size of the array: ";
    cin >> n;
    vector<int> arr(n);
    cout << "Enter elements of the array: ";
    for (int& num : arr) cin >> num;
    cout << "Enter the number to search: ";
    cin >> k;

    int result = searchNumber(arr, k);
    if (result != -1)
        cout << "First occurrence at position: " << result << endl;
    else
        cout << "Number not found in the array." << endl;

    return 0;
}
```

Output:



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code defines a function 'searchNumber' that iterates through a vector to find a target value. The main function prompts the user for the array size, elements, and the search value. The output window shows the user input: size 5, elements '1 5 2 9 0', search value 5, and the result 'First occurrence at position: 2'. A success message '=== Code Execution Successful ===' is also displayed.

```
main.cpp
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int searchNumber(vector<int>& arr, int k) {
6     for (int i = 0; i < arr.size(); i++) {
7         if (arr[i] == k) return i + 1; // 1-based indexing
8     }
9     return -1;
10 }
11
12 int main() {
13     int n, k;
14     cout << "Enter the size of the array: ";
15     cin >> n;
```

Output

```
Enter the size of the array: 5
Enter elements of the array: 1 5 2 9 0
Enter the number to search: 5
First occurrence at position: 2

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

2) Sorted Array Search

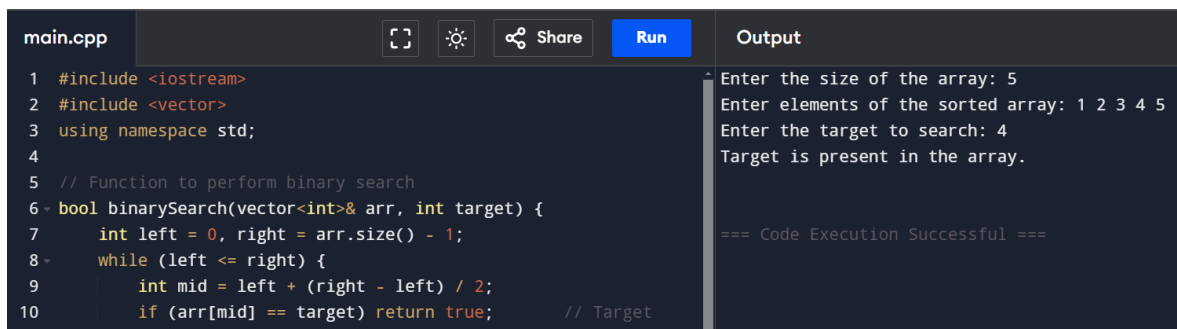
```
#include <iostream>
#include <iostream>
#include <vector>
using namespace std;
```

```
// Function to
perform binary
search
bool
binarySearch(vector<
int>& arr, int target) {
    int left = 0, right =
arr.size() - 1;
    while (left <= right)
    {
        int mid = left +
(right - left) / 2;
        if (arr[mid] ==
target) return true;
// Target found
        else if (arr[mid] <
target) left = mid + 1;
// Search in the right
half
        else right = mid -
1;
//
Search in the left half
    }
    return false; //
```

```
Target not found  
}
```

```
int main() {  
    int n, target;  
    cout << "Enter the  
size of the array: ";  
    cin >> n;  
  
    vector<int> arr(n);  
    cout << "Enter  
elements of the  
sorted array: ";  
    for (int& num : arr)  
        cin >> num;  
  
    cout << "Enter the  
target to search: ";  
    cin >> target;  
  
    if  
(binarySearch(arr,  
target))  
        cout << "Target  
is present in the  
array." << endl;  
    else  
        cout << "Target  
is not present in the  
array." << endl;  
  
    return 0;  
}
```

Output:



The screenshot shows a C++ IDE with a file named `main.cpp`. The code implements a binary search function. The output window shows the following text:

```
Enter the size of the array: 5
Enter elements of the sorted array: 1 2 3 4 5
Enter the target to search: 4
Target is present in the array.

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

3) Find First and Last Position of Element

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

// Function to find the first occurrence of the target

```
int findFirst(vector<int>& arr, int target) {
    int left = 0, right = arr.size() - 1, result = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) {
            result = mid;
            right = mid - 1; // Search in the left half for earlier occurrences
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return result;
}
```

// Function to find the last occurrence of the target

```
int findLast(vector<int>& arr, int target) {
    int left = 0, right = arr.size() - 1, result = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) {
            result = mid;
            left = mid + 1; // Search in the right half for later occurrences
        } else if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
}
```

```

    return result;
}

int main() {
    int n, target;
    cout << "Enter the size of the array: ";
    cin >> n;

    vector<int> arr(n);
    cout << "Enter elements of the sorted array: ";
    for (int& num : arr) cin >> num;

    cout << "Enter the target to find: ";
    cin >> target;

    int first = findFirst(arr, target);
    int last = findLast(arr, target);

    if (first != -1 && last != -1) {
        cout << "First occurrence at index: " << first << endl;
        cout << "Last occurrence at index: " << last << endl;
    } else {
        cout << "Target not found in the array." << endl;
    }

    return 0;
}

```

Output:

The screenshot shows a C++ IDE with a file named 'main.cpp'. The code implements a binary search algorithm to find the first and last occurrences of a target in a sorted array. The output window shows the following interaction:

```

Enter the size of the array: 4
Enter elements of the sorted array: 1 2 3 4
Enter the target to find: 3
First occurrence at index: 2
Last occurrence at index: 2

=== Code Execution Successful ===

```

4) Squares of a Sorted Array

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

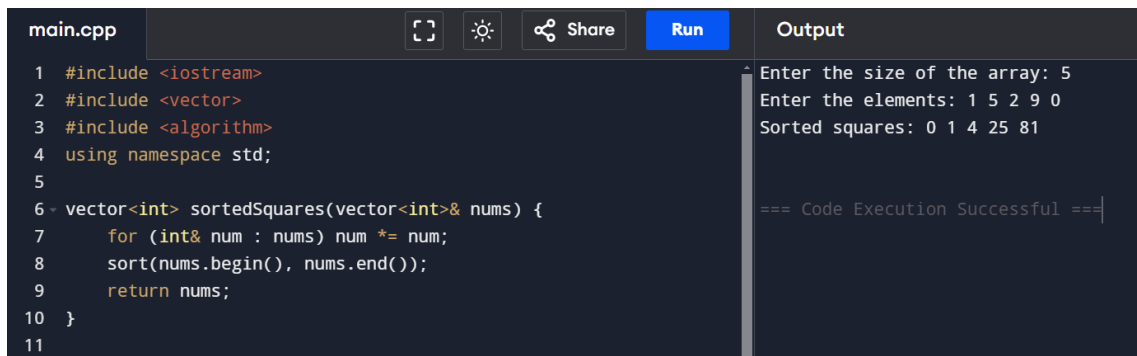
vector<int> sortedSquares(vector<int>& nums) {
    for (int& num : nums) num *= num;
    sort(nums.begin(), nums.end());
    return nums;
}

int main() {
    int n;
    cout << "Enter the size of the array: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter the elements: ";
    for (int& num : nums) cin >> num;

    vector<int> result = sortedSquares(nums);
    cout << "Sorted squares: ";
    for (int val : result) cout << val << " ";
    cout << endl;

    return 0;
}
```

Output:



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code defines a function 'sortedSquares' that takes a vector of integers and returns a new vector containing the squares of the elements, sorted in ascending order. The output window shows the program's execution: it prompts for the size of the array (5), then for the elements (1 5 2 9 0), and finally displays the sorted squares (0 1 4 25 81). The execution is successful.

```
main.cpp  [Icons]  Run  Output
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  vector<int> sortedSquares(vector<int>& nums) {
7      for (int& num : nums) num *= num;
8      sort(nums.begin(), nums.end());
9      return nums;
10 }
11
```

Enter the size of the array: 5
Enter the elements: 1 5 2 9 0
Sorted squares: 0 1 4 25 81
=== Code Execution Successful ===

5) Search in a 2D Matrix

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

```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int rows = matrix.size(), cols = matrix[0].size();
    int left = 0, right = rows * cols - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int midVal = matrix[mid / cols][mid % cols];
        if (midVal == target) return true;
        else if (midVal < target) left = mid + 1;
        else right = mid - 1;
    }
    return false;
}
```

```
int main() {
    int rows, cols, target;
    cout << "Enter rows and columns: ";
    cin >> rows >> cols;

    vector<vector<int>> matrix(rows, vector<int>(cols));
    cout << "Enter elements row by row:\n";
    for (auto& row : matrix) {
        for (int& val : row) cin >> val;
```

```

}

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

if (searchMatrix(matrix, target)) cout << "Target found.\n";
else cout << "Target not found.\n";

return 0;
}

```

Output:

main.cpp	Output
<pre> 1 #include <iostream> 2 #include <vector> 3 using namespace std; 4 5 bool searchMatrix(vector<vector<int>>& matrix, int target) { 6 int rows = matrix.size(), cols = matrix[0].size(); 7 int left = 0, right = rows * cols - 1; 8 9 while (left <= right) { 10 int mid = left + (right - left) / 2; 11 int midVal = matrix[mid / cols][mid % cols]; 12 if (midVal == target) return true; 13 else if (midVal < target) left = mid + 1; 14 else right = mid - 1; </pre>	<pre> Enter rows and columns: 3 4 Enter elements row by row: 1 3 5 7 10 11 16 20 23 30 34 60 Enter the target: 3 Target found. === Code Execution Successful === </pre>

6) Median of Two Sorted Arrays

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

double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    vector<int> merged(nums1.begin(), nums1.end());
    merged.insert(merged.end(), nums2.begin(), nums2.end());
    sort(merged.begin(), merged.end());
    int n = merged.size();
    if (n % 2 == 0) return (merged[n / 2 - 1] + merged[n / 2]) / 2.0;
    else return merged[n / 2];
}


int main() {
    int n1, n2;
    cout << "Enter size of first array: ";
    cin >> n1;
    vector<int> nums1(n1);
    cout << "Enter first array: ";
    for (int& num : nums1) cin >> num;

    cout << "Enter size of second array: ";
    cin >> n2;
    vector<int> nums2(n2);
    cout << "Enter second array: ";
    for (int& num : nums2) cin >> num;

    cout << "Median: " << findMedianSortedArrays(nums1, nums2) << endl;

    return 0;
}
```

Output :



The screenshot shows a C++ IDE with a file named `main.cpp`. The code defines a function `findMedianSortedArrays` that takes two sorted arrays and returns their median. The output window shows the program's execution with user input for array sizes and elements, resulting in a median of 3.5.

```
main.cpp
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 double findMedianSortedArrays(vector<int>& nums1, vector<int>&
  nums2) {
7     vector<int> merged(nums1.begin(), nums1.end());
8     merged.insert(merged.end(), nums2.begin(), nums2.end());
9     sort(merged.begin(), merged.end());
10    int n = merged.size();
11    if (n % 2 == 0) return (merged[n / 2 - 1] + merged[n / 2])
      / 2.0;
12    else return merged[n / 2];
}
```

Output

```
Enter size of first array: 5
Enter first array: 1 5 2 9 0
Enter size of second array: 5
Enter second array: 1 5 2 7 6
Median: 3.5

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

7) Merge K Sorted Lists

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

```
struct ListNode {
    int val;
    ListNode* next;
    ListNode(int x) : val(x), next(nullptr) {}
};
```

```
struct Compare {
    bool operator()(ListNode* a, ListNode* b) {
        return a->val > b->val;
    }
};
```

```
ListNode* mergeKLists(vector<ListNode*>& lists) {
    priority_queue<ListNode*, vector<ListNode*>, Compare> pq;
    for (auto node : lists) {
        if (node) pq.push(node);
    }
```

```
ListNode* dummy = new ListNode(0);
ListNode* tail = dummy;
```

```

while (!pq.empty()) {
    ListNode* curr = pq.top();
    pq.pop();
    tail->next = curr;
    tail = tail->next;
    if (curr->next) pq.push(curr->next);
}
return dummy->next;
}

```

// Helper functions to create and print a linked list

```

ListNode* createList(vector<int>& nums) {
    ListNode* head = nullptr, *tail = nullptr;
    for (int num : nums) {
        ListNode* newNode = new ListNode(num);
        if (!head) head = tail = newNode;
        else {
            tail->next = newNode;
            tail = newNode;
        }
    }
    return head;
}

```

```

void printList(ListNode* head) {
    while (head) {
        cout << head->val << " ";
        head = head->next;
    }
    cout << endl;
}

```

```

int main() {
    int k;
    cout << "Enter the number of linked lists: ";
    cin >> k;

    vector<ListNode*> lists(k);
    for (int i = 0; i < k; ++i) {
        int n;
        cout << "Enter size of list " << i + 1 << ": ";
        cin >> n;
        vector<int> nums(n);
        cout << "Enter elements: ";
    }
}

```

```

        for (int& num : nums) cin >> num;
        lists[i] = createList(nums);
    }

    ListNode* merged = mergeKLists(lists);
    cout << "Merged list: ";
    printList(merged);

    return 0;
}

```

Output :

main.cpp	Output
<pre> 1 #include <iostream> 2 #include <queue> 3 #include <vector> 4 using namespace std; 5 6 struct ListNode { 7 int val; 8 ListNode* next; 9 ListNode(int x) : val(x), next(nullptr) {} 10 }; 11 12 struct Compare { 13 bool operator()(ListNode* a, ListNode* b) { 14 return a->val > b->val; 15 } </pre>	<pre> Enter the number of linked lists: 3 Enter size of list 1: 3 Enter elements: 1 4 5 Enter size of list 2: 3 Enter elements: 1 3 4 Enter size of list 3: 2 Enter elements: 2 6 Merged list: 1 1 2 3 4 4 5 6 === Code Execution Successful === </pre>

8) Find Minimum in Rotated Sorted Array II

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

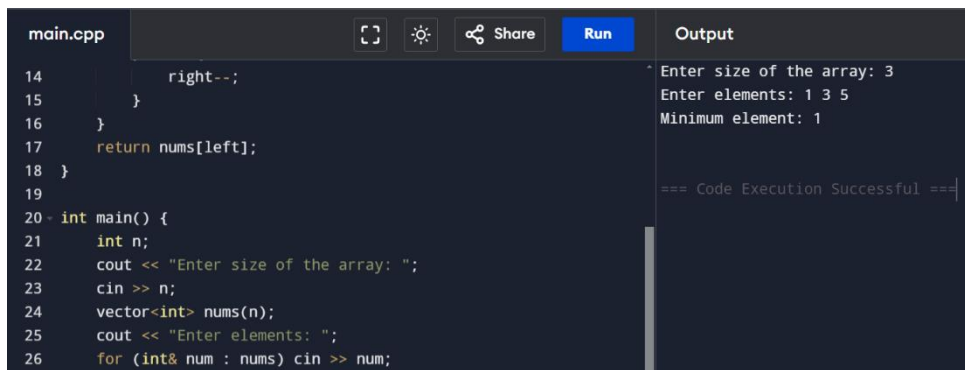
int findMin(vector<int>& nums) {
    int left = 0, right = nums.size() - 1;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] > nums[right]) {
            left = mid + 1;
        } else if (nums[mid] < nums[right]) {
            right = mid;
        } else {
            right--;
        }
    }
    return nums[left];
}

int main() {
    int n;
    cout << "Enter size of the array: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter elements: ";
    for (int& num : nums) cin >> num;

    cout << "Minimum element: " << findMin(nums) << endl;

    return 0;
}
```

Output:

A screenshot of a C++ IDE. The left pane shows a file named 'main.cpp' with C++ code. The code includes a recursive function 'right--' and a 'main' function that prompts the user for array size and elements, then calls 'sortEvenOdd'. The right pane shows the output of the program, which matches the user input: 'Enter size of the array: 3', 'Enter elements: 1 3 5', and 'Minimum element: 1'. Below the output, it says '=== Code Execution Successful ==='.

```
main.cpp
14     right--;
15 }
16 }
17 return nums[left];
18 }
19
20 int main() {
21     int n;
22     cout << "Enter size of the array: ";
23     cin >> n;
24     vector<int> nums(n);
25     cout << "Enter elements: ";
26     for (int& num : nums) cin >> num;
```

```
Output
Enter size of the array: 3
Enter elements: 1 3 5
Minimum element: 1

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

9) Sort Even and Odd Indices Independently

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

```
vector<int> sortEvenOdd(vector<int>& nums) {
    vector<int> even, odd;
    for (int i = 0; i < nums.size(); ++i) {
        if (i % 2 == 0) even.push_back(nums[i]);
        else odd.push_back(nums[i]);
    }
    sort(even.begin(), even.end());
    sort(odd.rbegin(), odd.rend());

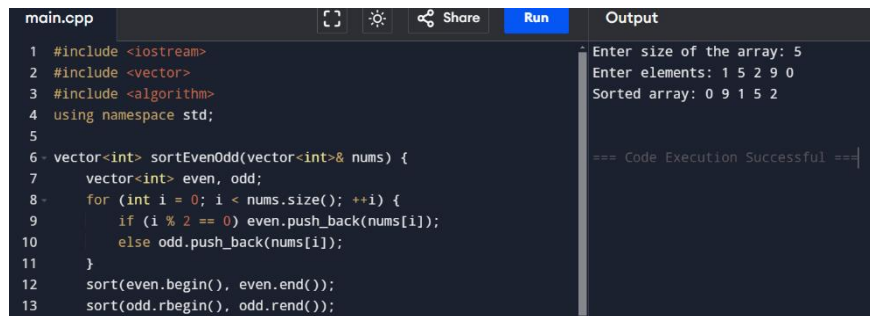
    vector<int> result(nums.size());
    for (int i = 0, e = 0, o = 0; i < nums.size(); ++i) {
        result[i] = (i % 2 == 0) ? even[e++] : odd[o++];
    }
    return result;
}
```

```
int main() {
    int n;
    cout << "Enter size of the array: ";
    cin >> n;
    vector<int> nums(n);
    cout << "Enter elements: ";
    for (int& num : nums) cin >> num;
```

```
vector<int> result = sortEvenOdd(nums);
cout << "Sorted array: ";
for (int num : result) cout << num << " ";
cout << endl;

return 0;
}
```

Output :



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code defines a function 'sortEvenOdd' that separates even and odd numbers from an input vector, sorts them, and returns the result. The 'Run' button is highlighted in blue. The 'Output' pane on the right shows the program's execution: it prompts for the array size (5), the elements (1 5 2 9 0), and displays the sorted array (0 9 1 5 2). A success message '=== Code Execution Successful ===' is also shown.

```
main.cpp  [ ]  [ ]  [ ]  Share  Run  Output
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 vector<int> sortEvenOdd(vector<int>& nums) {
7     vector<int> even, odd;
8     for (int i = 0; i < nums.size(); ++i) {
9         if (i % 2 == 0) even.push_back(nums[i]);
10        else odd.push_back(nums[i]);
11    }
12    sort(even.begin(), even.end());
13    sort(odd.rbegin(), odd.rend());
14
15    vector<int> result;
16    result.insert(result.end(), even.begin(), even.end());
17    result.insert(result.end(), odd.rbegin(), odd.rend());
18    return result;
19 }
```

Enter size of the array: 5
Enter elements: 1 5 2 9 0
Sorted array: 0 9 1 5 2

=== Code Execution Successful ===

10) Sorted GCD Pair Queries

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

// Function to calculate all GCD pairs and sort them
vector<int> calculateSortedGCDPairs(vector<int>& nums) {
    vector<int> gcdPairs;
    int n = nums.size();

    // Calculate GCD for all pairs (nums[i], nums[j]) where i < j
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            gcdPairs.push_back(gcd(nums[i], nums[j]));
        }
    }

    // Sort the GCD pairs
    sort(gcdPairs.begin(), gcdPairs.end());
    return gcdPairs;
}

// Function to handle the queries
vector<int> processQueries(vector<int>& gcdPairs, vector<int>& queries) {
    vector<int> results;
    for (int q : queries) {
        if (q >= 0 && q < gcdPairs.size()) {
            results.push_back(gcdPairs[q]);
        } else {
            results.push_back(-1); // Invalid query index
        }
    }
    return results;
}

int main() {
    int n, q;
    cout << "Enter the size of the array: ";
    cin >> n;

    vector<int> nums(n);
```



```

cout << "Enter elements of the array: ";
for (int& num : nums) cin >> num;

cout << "Enter the number of queries: ";
cin >> q;

vector<int> queries(q);
cout << "Enter query indices: ";
for (int& query : queries) cin >> query;

// Calculate sorted GCD pairs
vector<int> gcdPairs = calculateSortedGCDPairs(nums);

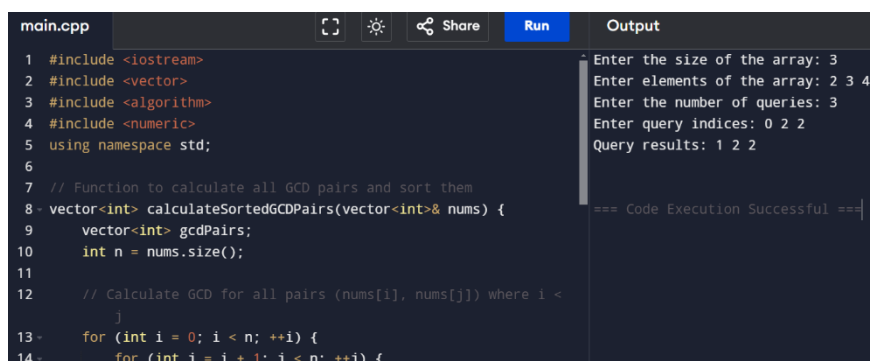
// Process the queries
vector<int> results = processQueries(gcdPairs, queries);

// Output the results
cout << "Query results: ";
for (int res : results) cout << res << " ";
cout << endl;

return 0;
}

```

Output :



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code is a program that calculates the GCD of all pairs in an array and processes queries. The output window shows the following interaction:

```

Enter the size of the array: 3
Enter elements of the array: 2 3 4
Enter the number of queries: 3
Enter query indices: 0 2 2
Query results: 1 2 2

```

At the bottom of the output window, it says "=== Code Execution Successful ===".

