



**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

**Company Name:** Adobe

### **Easy Question 1**

#### **Code:**

```
class Solution {  
public:  
    string reverseWordsRec(const string& s, int start = 0)  
    {  
        int n = s.size();  
        while (start < n && s[start] == '.')  
        {  
            start++;  
        }  
        if (start == n) return "";  
        int end = start;  
        while (end < n && s[end] != '.') {  
            end++;  
        }  
        string currentWord = s.substr(start, end - start);  
        string rest = reverseWordsRec(s, end);  
  
        if (rest.empty()) {  
            return currentWord;  
        } else {  
            return rest + "." + currentWord;  
        }  
    }  
    string reverseWords(string s) {  
        return reverseWordsRec(s, 0);  
    }  
};
```



**Marwari University**  
**Faculty of Technology**  
**Department of Information and Communication Technology**

**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

## Output:

The screenshot shows a browser window with the GeeksforGeeks website. The URL is [geeksforgeeks.org/problems/reverse-words-in-a-given-string5459/1](https://www.geeksforgeeks.org/problems/reverse-words-in-a-given-string5459/1). The page displays a solved problem with the following details:

- Problem Solved Successfully**
- Test Cases Passed:** 1111 / 1111
- Attempts:** Correct / Total: 2 / 3
- Accuracy:** 66%
- Time Taken:** 0.05
- Note:** You get marks only for the first correct submission if you solve the problem without viewing the full solution.
- Suggested Contests:** Based on your excellent performance, we believe you are fully prepared to participate in this upcoming contest.

The right side of the screen shows the C++ code for the solution:

```
1- class Solution {
2 public:
3     string reverseWordsRec(const string& s, int start = 0)
4     {
5         int n = s.size();
6         while (start < n && s[start] == '.')
7         {
8             start++;
9         }
10        if (start == n) return "";
11        int end = start;
12        while (end < n && s[end] != '.') {
13            end++;
14        }
15
16        string currentWord = s.substr(start, end - start);
17        string rest = reverseWordsRec(s, end);
18
19        if (rest.empty()) {
20            return currentWord;
21        } else {
22            return rest + "." + currentWord;
23        }
24    }
25
26    string reverseWords(string s) {
27        return reverseWordsRec(s, 0);
28    }
29};
30};
```

## Easy Question 2

### Code:

```
class Solution {
public:
    bool hasPathSum(Node* root, int targetSum) {
        if (root == nullptr) {
            return false;
        }

        if (!root->left && !root->right) {
            return targetSum == root->data;
        }

        return hasPathSum(root->left, targetSum - root->data) ||
               hasPathSum(root->right, targetSum - root->data);
    }
};
```



**Marwari University**  
**Faculty of Technology**  
**Department of Information and Communication Technology**

**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

## Output:

The screenshot shows a browser window with the URL [geeksforgeeks.org/problems/root-to-leaf-path-sum/1](https://geeksforgeeks.org/problems/root-to-leaf-path-sum/1). The page displays a solved problem with the following details:

- Test Cases Passed:** 1115 / 1115
- Attempts : Correct / Total:** 1 / 3
- Accuracy :** 33%
- Points Scored :** 2 / 2
- Time Taken:** 0.05
- Your Total Score:** 4

The code submitted is:

```
1- class Solution {  
2- public:  
3-     bool hasPathSum(Node* root, int targetSum) {  
4-         if (root == nullptr) {  
5-             return false;  
6-         }  
7-         if (!root->left && !root->right) {  
8-             return targetSum == root->data;  
9-         }  
10-        return hasPathSum(root->left, targetSum - root->data) ||  
11-               hasPathSum(root->right, targetSum - root->data);  
12-    }  
13-};  
14-  
15-};  
16-
```

Below the code editor, there are links for "Right View of Binary Tree", "K distance from root", and "Mirror Tree". A "Suggested Contest" section is also visible.

## Easy Question 3

**Code:**

```
class Solution {  
public:
```

```
    bool isBSTUtil(Node* node, int minValue, int maxValue) {  
        if (node == nullptr) {  
            return true;  
        }  
        if (node->data <= minValue || node->data >= maxValue) {  
            return false;  
        }  
        return isBSTUtil(node->left, minValue, node->data) &&  
               isBSTUtil(node->right, node->data, maxValue);  
    }  
    bool isBST(Node* root) {  
        const int MIN_VALUE = 0;  
        const int MAX_VALUE = 1000000001;  
        return isBSTUtil(root, MIN_VALUE, MAX_VALUE);  
    }
```



**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

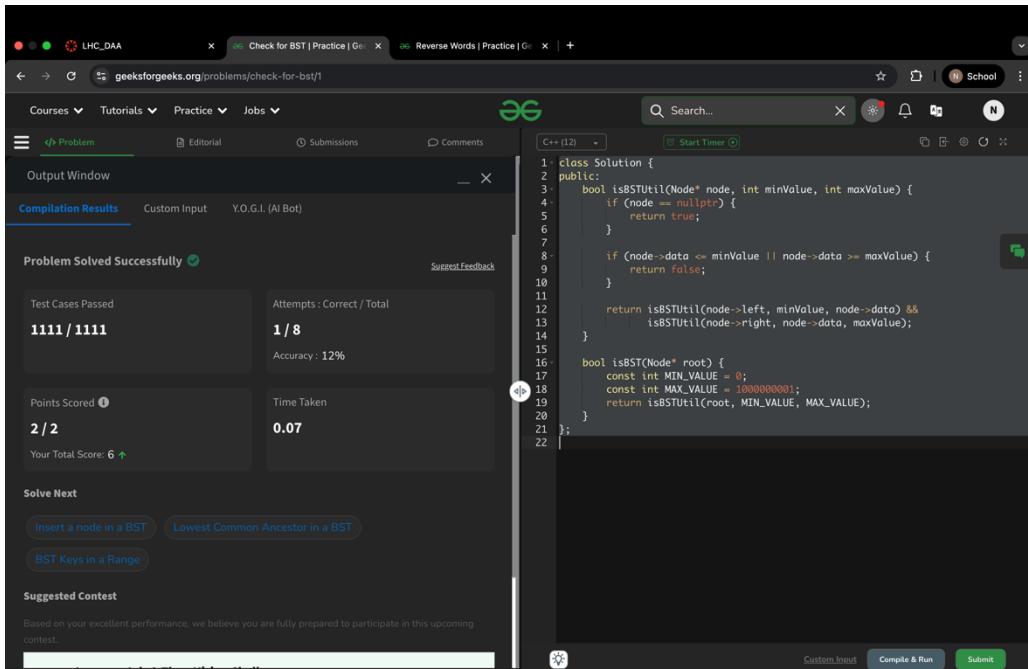
**Date: 02-08-2025**

**Enrollment No: 92301733054**

}

};

## Output:



The screenshot shows a successful submission for a problem titled "Check for BST". The user has solved the problem successfully, with all 1111 test cases passed. The code submitted is a C++ solution for checking if a binary search tree is valid. The code uses a recursive helper function `isBSTUtil` and a main function `isBST` to validate the tree's properties.

```
class Solution {
public:
    bool isBSTUtil(Node* node, int minValue, int maxValue) {
        if (node == nullptr) {
            return true;
        }

        if (node->data <= minValue || node->data >= maxValue) {
            return false;
        }

        return isBSTUtil(node->left, minValue, node->data) &&
               isBSTUtil(node->right, node->data, maxValue);
    }

    bool isBST(Node* root) {
        const int MIN_VALUE = 0;
        const int MAX_VALUE = 100000000;
        return isBSTUtil(root, MIN_VALUE, MAX_VALUE);
    }
};
```

## Easy Question 4

**Code:**

class Solution

{

public:

int search(vector<int>& arr, int key)

{

int low = 0, high = arr.size() - 1;

while (low <= high)

{

int mid = low + (high - low) / 2;

if (arr[mid] == key)

    return mid;

if (arr[low] <= arr[mid])

{

    if (arr[low] <= key && key < arr[mid])



**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

```
high = mid - 1;  
else  
    low = mid + 1;  
}  
else {  
    if (arr[mid] < key && key <= arr[high])  
        low = mid + 1;  
    else  
        high = mid - 1;  
}  
return -1;  
}  
};
```

## Output:

The screenshot shows a browser window for geeksforgeeks.org/problems/search-in-a-rotated-array4618/1. The page displays a solved problem with the following details:

- Courses**, **Tutorials**, **Practice**, **Jobs** dropdown menus.
- Output Window** button.
- Compilation Results** section: Problem Solved Successfully, Test Cases Passed 1116 / 1116, Attempts : Correct / Total 2 / 5, Accuracy : 40%.
- Time Taken**: 0.16.
- Note**: You get marks only for the first correct submission if you solve the problem without viewing the full solution.
- Solve Next** buttons: Square Root, Counting elements in two arrays, Smallest Positive Missing.
- Suggested Contest** message: Based on your excellent performance, we believe you are fully prepared to participate in this upcoming contest.
- Code Editor** (C++ 12):

```
1 class Solution
2 {
3     public:
4         int search(vector<int>& arr, int key)
5     {
6         int low = 0, high = arr.size() - 1;
7         while (low <= high)
8         {
9             int mid = low + (high - low) / 2;
10            if (arr[mid] == key)
11                return mid;
12            if (arr[low] <= arr[mid])
13            {
14                if (arr[low] <= key && key < arr[mid])
15                    high = mid - 1;
16                else
17                    low = mid + 1;
18            }
19            else {
20                if (arr[mid] < key && key <= arr[high])
21                    low = mid + 1;
22                else
23                    high = mid - 1;
24            }
25        }
26        return -1;
27    }
28};
```
- Toolbar**: Start Timer, Compile & Run, Submit.



**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

### **Easy Question 5**

**Code:**

```
class Solution {  
public:  
    Node* reverseList(Node* head) {  
        Node* prev = NULL;  
        Node* curr = head;  
        Node* next = NULL;  
  
        while (curr != NULL) {  
            next = curr->next;  
            curr->next = prev;  
            prev = curr;  
            curr = next;  
        }  
        return prev;  
    }  
};
```



**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

## **Output:**

The screenshot shows a successful submission on the GeeksforGeeks platform. The code provided is:

```
1- class Solution {
2- public:
3-     Node* reverseList(Node* head) {
4-         Node* prev = NULL;
5-         Node* curr = head;
6-         Node* next = NULL;
7-
8-         while (curr != NULL) {
9-             next = curr->next;
10-            curr->next = prev;
11-            prev = curr;
12-            curr = next;
13-        }
14-        return prev;
15-    }
16-};
17-
```

Test Cases Passed: 1115 / 1115. Attempts: Correct / Total: 1 / 1. Accuracy: 100%. Points Scored: 2 / 2. Time Taken: 0.12. Your Total Score: 12.

## **Easy Question 6**

**Code:**

```
class Solution {
public:
    vector<int> findSpiral(Node* root) {
        vector<int> result;
        if (!root) return result;
        stack<Node*> s1;
        stack<Node*> s2;
        s1.push(root);
        while (!s1.empty() || !s2.empty()) {
            while (!s1.empty()) {
                Node* curr = s1.top();
                s1.pop();
                result.push_back(curr->data);
                if (curr->right) s2.push(curr->right);
            }
            while (!s2.empty()) {
                Node* curr = s2.top();
                s2.pop();
                result.push_back(curr->data);
                if (curr->left) s1.push(curr->left);
            }
        }
        return result;
    }
};
```



**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**

```
        if (curr->left) s2.push(curr->left);
    }
    while (!s2.empty()) {
        Node* curr = s2.top();
        s2.pop();
        result.push_back(curr->data);
        if (curr->left) s1.push(curr->left);
        if (curr->right) s1.push(curr->right);
    }
}
return result;
};

};
```

## OUTPUT:

The screenshot shows a successful submission on the GeeksforGeeks platform. The code has been compiled and run successfully, with all test cases passed. The user has solved the problem and earned 2/2 points, with a total score of 14.

```
1 class Solution {
2 public:
3     vector<int> findSpiral(Node* root) {
4         vector<int> result;
5         if (!root) return result;
6
7         stack<Node*> s1;
8         stack<Node*> s2;
9
10        s1.push(root);
11
12        while (!s1.empty() || !s2.empty()) {
13            while (!s1.empty()) {
14                Node* curr = s1.top();
15                s1.pop();
16                result.push_back(curr->data);
17
18                if (curr->right) s2.push(curr->right);
19                if (curr->left) s2.push(curr->left);
20            }
21
22            while (!s2.empty()) {
23                Node* curr = s2.top();
24                s2.pop();
25                result.push_back(curr->data);
26
27                if (curr->left) s1.push(curr->left);
28                if (curr->right) s1.push(curr->right);
29            }
30
31        }
32
33        return result;
34    };
35}
```



**Marwadi  
University**

**Marwari University  
Faculty of Technology  
Department of Information and Communication Technology**

**Subject: Design and Analysis  
of Algorithms (01CT0512)**

**Name: Nikhil Bhanderi**

**Class: 5EK1--C**

**Long Hour Coding**

**Date: 02-08-2025**

**Enrollment No: 92301733054**