











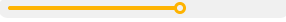
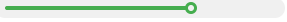
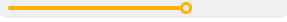
Preet Monga

Test ID: 354331085728244 | 7082165005 | 22bdo10062@cuchd.in

Test Date: December 9, 2024

Computer Science 88 /100 	Logical Ability 64 /100 	Computer Programming 63 /100 	Quantitative Ability (Advanced) 63 /100 
English Comprehension 65 /100 	WriteX - Essay Writing 77 /100 	Automata Fix 72 /100 	Automata Pro 36 /100 
Personality  Completed			

Computer Science  88 / 100		
OS and Computer Architecture  80 / 100	DBMS  100 / 100	Computer Networks  83 / 100

Logical Ability  64 / 100		
Inductive Reasoning  62 / 100	Deductive Reasoning  67 / 100	Abductive Reasoning  64 / 100

Computer Programming

63 / 100

Basic Programming

62 / 100

Data Structures

65 / 100

OOP and Complexity Theory

62 / 100

Quantitative Ability (Advanced)

63 / 100

Basic Mathematics

58 / 100

Advanced Mathematics

64 / 100

Applied Mathematics

68 / 100

English Comprehension

65 / 100

CEFR: **B2**

Grammar

68 / 100

Vocabulary

61 / 100

Comprehension

66 / 100

WriteX - Essay Writing

77 / 100

CEFR: **C1**

Content Score

74 / 100

Grammar Score

85 / 100

Automata Fix

72 / 100

Logical Error

75 / 100

Code Reuse

50 / 100

Syntactical Error

100 / 100

Automata Pro

36 / 100

Programming Ability

40 / 100

Programming Practices

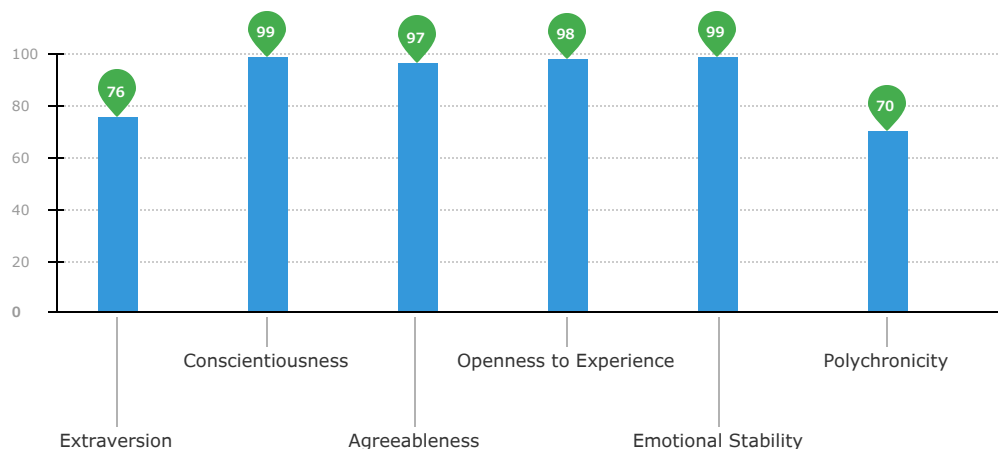
50 / 100

Functional Correctness

29 / 100

Personality

Completed



Competencies

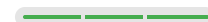
People Interaction



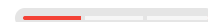
Self-Drive



Trainability



Repetitive Job Suitability



Work attributes

1 | Introduction

About the Report

This report provides a detailed analysis of the candidate's performance on different assessments. The tests for this job role were decided based on job analysis, O*Net taxonomy mapping and/or criterion validity studies. The candidate's responses to these tests help construct a profile that reflects her/his likely performance level and achievement potential in the job role

This report has the following sections:

The **Summary** section provides an overall snapshot of the candidate's performance. It includes a graphical representation of the test scores and the subsection scores.

The **Insights** section provides detailed feedback on the candidate's performance in each of the tests. The descriptive feedback includes the competency definitions, the topics covered in the test, and a note on the level of the candidate's performance.

The **Response** section captures the response provided by the candidate. This section includes only those tests that require a subjective input from the candidate and are scored based on artificial intelligence and machine learning.

The **Learning Resources** section provides online and offline resources to improve the candidate's knowledge, abilities, and skills in the different areas on which s/he was evaluated.

Score Interpretation

All the test scores are on a scale of 0-100. All the tests except personality and behavioural evaluation provide absolute scores. The personality and behavioural tests provide a norm-referenced score and hence, are percentile scores. Throughout the report, the colour codes used are as follows:

- Scores between 67 and 100
- Scores between 33 and 67
- Scores between 0 and 33

2 | Insights

English Comprehension

65 / 100

CEFR: **B2**

This test aims to measure your vocabulary, grammar and reading comprehension skills.

You have a good understanding of commonly used grammatical constructs. You are able to read and understand articles, reports and letters/emails related to your day-to-day work. The ability to read, understand and interpret business-related documents is essential in most jobs, especially the ones that involve research, technical reading and content writing.

Logical Ability

64 / 100



Inductive Reasoning

62 / 100

This competency aims to measure the your ability to synthesize information and derive conclusions.

It is commendable that you have excellent inductive reasoning skills. You are able to make specific observations to generalize situations and also formulate new generic rules from variable data.



Deductive Reasoning

67 / 100

This competency aims to measure the your ability to synthesize information and derive conclusions.

It is commendable that you have excellent inductive reasoning skills. You are able to make specific observations to generalize situations and also formulate new generic rules from variable data.



Abductive Reasoning

64 / 100

Quantitative Ability (Advanced)

63 / 100

This test aims to measure your ability to solve problems on basic arithmetic operations, probability, permutations and combinations, and other advanced concepts.

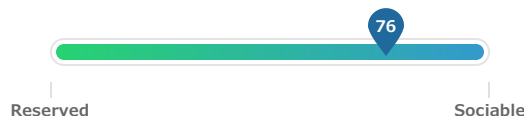
You are able to solve word problems on basic concepts of percentages, ratio, proportion, interest, time and work. Having a strong hold on these concepts can help you understand the concept of work efficiency and how interest is accrued on bank savings. It can also guide you in time management, work planning, and resource allocation in complex projects.

Personality

Competencies



Extraversion



Extraversion refers to a person's inclination to prefer social interaction over spending time alone. Individuals with high levels of extraversion are perceived to be outgoing, warm and socially confident.

- You are outgoing and seek out opportunities to meet new people.
- You tend to enjoy social gatherings and feels comfortable amongst strangers and friends equally.
- You display high energy levels and like to indulge in thrilling and exciting activities.
- You may tend to be assertive about your opinions and prefer action over contemplation.
- You take initiative and are more inclined to take charge than to wait for others to lead the way.
- Your personality is well suited for jobs demanding frequent interaction with people.



Conscientiousness

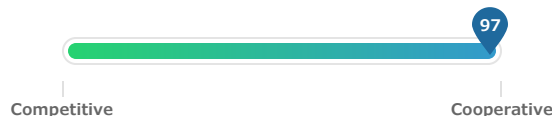


Conscientiousness is the tendency to be organized, hard working and responsible in one's approach to your work. Individuals with high levels of this personality trait are more likely to be ambitious and tend to be goal-oriented and focused.

- You value order and self discipline and tends to pursue ambitious endeavours.
- You believe in the importance of structure and is very well-organized.
- You carefully review facts before arriving at conclusions or making decisions based on them.
- You strictly adhere to rules and carefully consider the situation before making decisions.
- You tend to have a high level of self confidence and do not doubt your abilities.
- You generally set and work toward goals, try to exceed expectations and are likely to excel in most jobs, especially those which require careful or meticulous approach.



Agreeableness



Agreeableness refers to an individual's tendency to be cooperative with others and it defines your approach to interpersonal relationships. People with high levels of this personality trait tend to be more considerate of people around them and are more likely to work effectively in a team.

- You are considerate and sensitive to the needs of others.
- You tend to put the needs of others ahead of your own.
- You are likely to trust others easily without doubting their intentions.
- You are compassionate and may be strongly affected by the plight of both friends and strangers.
- You are humble and modest and prefer not to talk about personal accomplishments.
- Your personality is more suitable for jobs demanding cooperation among employees.



Openness to Experience



Openness to experience refers to a person's inclination to explore beyond conventional boundaries in different aspects of life. Individuals with high levels of this personality trait tend to be more curious, creative and innovative in nature.

- You tend to be curious in nature and is generally open to trying new things outside your comfort zone.
- You may have a different approach to solving conventional problems and tend to experiment with those solutions.
- You are creative and tends to appreciate different forms of art.
- You are likely to be in touch with your emotions and is quite expressive.
- Your personality is more suited for jobs requiring creativity and an innovative approach to problem solving.



Emotional Stability



Emotional stability refers to the ability to withstand stress, handle adversity, and remain calm and composed when working through challenging situations. People with high levels of this personality trait tend to be more in control of their emotions and are likely to perform consistently despite difficult or unfavourable conditions.

- You are calm and composed in nature.
- You tend to maintain composure during high pressure situations.
- You are very confident and comfortable being yourself.
- You find it easy to resist temptations and practice moderation.
- You are likely to remain emotionally stable in jobs with high stress levels.



Polychronicity



Polychronicity refers to a person's inclination to multitask. It is the extent to which the person prefers to engage in more than one task at a time and believes that such an approach is highly productive. While this trait describes the personality disposition of a person to multitask, it does not gauge their ability to do so successfully.

- You pursue multiple tasks simultaneously, switching between them when needed.
- You prefer working to achieve some progress on multiple tasks simultaneously than completing one task before moving on to the next task.
- You tend to believe that multitasking is an efficient way of doing things and prefers an action packed work life with multiple projects.

3 | Response

Automata Pro



36 / 100

[Code Replay](#)

Question 1 (Language: C++)

The computer systems of N employees of a company are arranged in a row. A technical fault in the power supply has caused some of the systems to turn OFF while the others remain ON. The employees whose systems are OFF are unable to work. The company does not like to see its employees sitting idle. So until the technical team can find the actual cause of the breakdown, the technical head has devised a temporary workaround for the OFF systems at a minimum cost. They decide to connect all the OFF systems to the nearest ON system with the shortest possible length of cable. To make this happen, they calculate the distance of each system from the first system.

Write an algorithm to help the technical head find the minimum length of cable they need to turn all the systems ON.

Scores

Programming Ability

 **60** / 100

Basic program structure is consistent. Right control structures exist with a few/partially correct data dependencies.

Functional Correctness

 **38** / 100

Partially correct basic functionality. The source code compiles and passes only some of the basic test cases. Some advanced or edge cases may randomly pass.

Programming Practices

 **100** / 100

High readability, high on program structure. The source code is readable and does not consist of any significant redundant/improper coding constructs.

Final Code Submitted

Compilation Status: Pass

```
1 // Header Files
2 #include<iostream>
3 #include<string>
4 #include<vector>
5 using namespace std;
6
7
8 /*
9 *
10 */
11 int minLength (vector<int> systemState, vector<int> dist)
12 {
13     int answer = 0;
```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code: O(N)

*N represents number of systems

Errors/Warnings

There are no errors in the candidate's code.


```

14 // Write your code here
15
16
17 if(systemState[0] == 0){
18     if(dist.size() > 1){
19         answer += (dist[1] - dist[0]);
20     }
21     systemState[0] = 1;
22 }
23 if(systemState[systemState.size() - 1] == 0){
24     answer += (dist[systemState.size() - 1] - dist[systemState.size() -
25 2]);
26     systemState[systemState.size() - 1] = 1;
27 }
28
29 int flag = 1;
30
31 while(flag == 1)
32 {
33     flag = 1;
34
35     for(int i = 1; i < systemState.size() - 1; i++)
36     {
37         if(systemState[i] == 0)
38         {
39             if(systemState[i+1] != 0 && systemState[i-1] != 0)
40             {
41
42                 int right = dist[i+1] - dist[i];
43                 int left = dist[i] - dist[i-1];
44                 answer += (min(left, right));
45                 systemState[i] = 1;
46             }
47
48             else if(systemState[i+1] != 0 && systemState[i-1] == 0)
49             {
50                 answer += dist[i+1] - dist[i];
51                 systemState[i] = 1;
52             }
53             else if(systemState[i+1] == 0 && systemState[i-1] != 0)
54             {
55                 answer += dist[i] - dist[i-1];
56                 systemState[i] = 1;
57             }
58         }
59
60         flag = 0;
61     }
62

```

Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

```

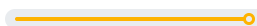
63 // }
64
65 }
66
67 return answer;
68 }
69
70 int main()
71 {
72
73 //input for systemState
74 int systemState_size;
75 cin >> systemState_size;
76 vector<int> systemState;
77 for ( int idx = 0; idx < systemState_size; idx++ )
78 {
79     int temp;
80     cin >> temp;
81     systemState.push_back(temp);
82 }
83
84 //input for dist
85 int dist_size;
86 cin >> dist_size;
87 vector<int> dist;
88 for ( int idx = 0; idx < dist_size; idx++ )
89 {
90     int temp;
91     cin >> temp;
92     dist.push_back(temp);
93 }
94
95 int result = minLength(systemState, dist);
96 cout << result;
97
98
99 return 0;
100 }
101

```

Test Case Execution

Passed TC: **52.38%**

Total score



11/21

88%

Basic(7/8)

22%

Advance(2/9)

50%

Edge(2/4)

Compilation Statistics

30

Total attempts

18

Successful

12

Compilation errors

0

Sample failed

5

Timed out

0

Runtime errors

Response time:

00:40:02

Average time taken between two compile attempts:

00:01:20

Average test case pass percentage per compile:

7.46%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

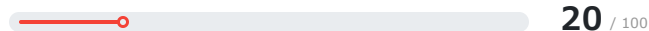
Question 2 (Language: C++)

A person is going to a book fair where all the books are star-rated. As they are interested in just two types of books, Horror and Sci-fi, they would buy the books from these two categories only. They would want to buy at least one book from each category so that the total star-rating of their books is the maximum. Also, the total price of the books should not exceed the amount of money that they can spend. The output is -1 if it is not possible to buy at least one book from both the categories with the money that they have.

Write an algorithm to help the person buy the books from both the categories.

Scores

Programming Ability



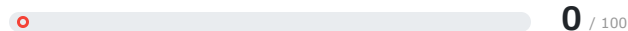
Code seems to be unrelated to the given problem.

Functional Correctness



The source code does not pass any basic test cases. It is either due to incorrect logic or runtime errors. Some advanced or edge cases may randomly pass.

Programming Practices



Programming practices score cannot be generated. This is because source code has syntax/runtime errors and is unparseable or the source code does not meet the minimum code-length specifications.

Final Code Submitted

Compilation Status: Pass

```

1 // Header Files
2 #include<iostream>
3 #include<string>
4 #include<vector>
5 using namespace std;
6
7
8 /*
9  * amount representing the amount of money Sheldon can spend.
10 horrorBooks is a grid where each row represents the star-rating and the price of a Horror book.
11 sciFiBooks is a grid where each row represents the star-rating and the price of a Sci-Fi book.
12 */
13 int maxRatingBooks(int amount, vector<vector<int>> horrorBooks, vector<vector<int>> sciFiBooks)
14 {
15     int answer = 0;
16     // Write your code here
17
18
19
20     return answer;
21 }
22
23 int main()
24 {
25     //input for amount
26     int amount;
27     cin >> amount;
28

```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code: $O(N^2)$

*N represents number of Horror books/Sci-fi books

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

```

29 //input for horrorBooks
30 int horrorBooks_row;
31 int horrorBooks_col;
32 cin >> horrorBooks_row;
33 cin >> horrorBooks_col;
34
35 vector<vector<int> > horrorBooks;
36 for ( int idx = 0; idx < horrorBooks_row; idx++ )
37 {
38     vector<int> temp_vector;
39     for ( int jdx = 0; jdx < horrorBooks_col; jdx++ )
40     {
41         int temp;
42         cin >> temp;
43         temp_vector.push_back(temp);
44     }
45     horrorBooks.push_back(temp_vector);
46 }
47
48 //input for sciFiBooks
49 int sciFiBooks_row;
50 int sciFiBooks_col;
51 cin >> sciFiBooks_row;
52 cin >> sciFiBooks_col;
53
54 vector<vector<int> > sciFiBooks;
55 for ( int idx = 0; idx < sciFiBooks_row; idx++ )
56 {
57     vector<int> temp_vector;
58     for ( int jdx = 0; jdx < sciFiBooks_col; jdx++ )
59     {
60         int temp;
61         cin >> temp;
62         temp_vector.push_back(temp);
63     }
64     sciFiBooks.push_back(temp_vector);
65 }
66
67
68 int result = maxRatingBooks(amount, horrorBooks, sciFiBooks);
69 cout << result;
70
71
72 return 0;
73 }
74

```

Test Case Execution

Passed TC: 0%

Total score

0/19

0%
Basic(0/8)

0%
Advance(0/8)

0%
Edge(0/3)

Compilation Statistics

0
Total attempts

0
Successful

0
Compilation errors

0
Sample failed

0
Timed out

0
Runtime errors

Response time:

00:07:02

Average time taken between two compile attempts:

00:00:00

Average test case pass percentage per compile:

0%

i

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

i

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

Automata Fix

72 / 100

Code Replay

Question 1 (Language: C++)

The function/method ***descendingSortArray*** performs an in-place sort on the given input list which will be sorted in descending order.

The function/method ***descendingSortArray*** accepts two arguments - *len*, an integer representing the length of the input list and *arr*, a list of integers representing the input list, respectively.

The function/method ***descendingSortArray*** compiles successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

Scores

Final Code Submitted

Compilation Status: Pass

```
1 // You can print the values to stdout for debugging
2 void descendingSortArray(int len, int* arr)
3 {
4     int i, j, temp;
5     for(i=0; i<=len-1;i++)
6     {
7         for(j=i; j<len;j++)
8         {
9             temp = 0;
10            if(arr[i]<arr[j])
11            {
12                temp=arr[i];
13                arr[i]=arr[j];
14                arr[j]=temp;
15            }
16        }
17    }
18 }
```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code:

*N represents

Errors/Warnings

There are no errors in the candidate's code.


Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 100%

Total score

 10/10

100%

Basic(6/6)

100%

Advance(4/4)

0%

Edge(0/0)

Compilation Statistics

3

Total attempts

3

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:01:00

Average time taken between two compile attempts:

00:00:20

Average test case pass percentage per compile:

70%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

Question 2 (Language: C++)

Lisa always forgets her birthday which is on the 5th of July. So, develop a function/method which will be helpful to remember her birthday.

The function/method **checkBirthDay** return an integer '1' if it is her birthday else returns 0. The function/method **checkBirthDay** accepts two arguments - *month*, a string representing the month of her birthday and *day*, an integer representing the date of her birthday.

The function/method **checkBirthDay** compiles successfully but fails to return the desired result for some test cases. Your task is to fix the code so that it passes all the test cases.

Scores

Final Code Submitted

Compilation Status: Pass

```
1 // You can print the values to stdout for debugging
2 #include<string.h>
3 using namespace std;
4 int checkBirthDay(char* month, int day)
5 {
6     if((strcmp(month,"July")) && (day==5))
7         return 1;
8     else
9         return 0;
10 }
11
```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code:

*N represents

Errors/Warnings

There are no errors in the candidate's code.

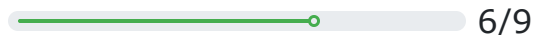
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: **66.67%**

Total score



67%

Basic(4/6)

67%

Advance(2/3)

0%

Edge(0/0)

Compilation Statistics

3

Total attempts

3

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:00:39

Average time taken between two compile attempts:

00:00:13

Average test case pass percentage per compile:

48.1%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

Question 3 (Language: C++)

You are given a predefined structure Point and also a collection of related functions/methods that can be used to perform some basic operations on the structure.

You must implement the function/method ***isTriangle*** which accepts three points *P1*, *P2*, *P3* as inputs and checks whether the given three points form a triangle.

If they form a triangle, the function/method returns an integer 1. Otherwise, it returns an integer 0.

Helper Description

The following class is used to represent point and is already implemented in the default code (Do not write these definitions again in your code):

```
class Point
{
    private:
        int X;
        int Y;

        double Point_calculateDistance(Point *point1, Point *point2)
        {
            /*Return the euclidean distance between two input points.
            This can be called as -
            * If P1 and P2 are two points then -
            * P1->Point_calculateDistance(P2);*/
        }
}
```

Scores

Final Code Submitted

Compilation Status: Pass

```
1 // You can print the values to stdout for debugging
2 using namespace std;
3 int isTriangle(Point *P1, Point *P2, Point *P3)
4 {
5     // write your code here
6     int len1 = P1->Point_calculateDistance(P2);
7     int len2 = P1->Point_calculateDistance(P3);
8     int len3 = P2->Point_calculateDistance(P3);
9 }
```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code:

*N represents

```

10 int collinear = 0;
11
12
13 if((len1 + len2) == len3 || (len2 + len3) == len1 || (len1 + len3) == len2)
14 {
15     return 0;
16 }
17
18
19 if((len1 + len2) > len3 || (len2 + len3) > len1 || (len1 + len3) > len2)
20 {
21     return 1;
22 }
23
24 return 0;
25 }

```

Errors/Warnings

There are no errors in the candidate's code.

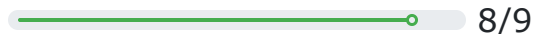
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: **88.89%**

Total score



83%

Basic(5/6)

100%

Advance(2/2)

100%

Edge(1/1)

Compilation Statistics

17

Total attempts

8

Successful

9

Compilation errors

6

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:12:18

Average time taken between two compile attempts:

00:00:43

Average test case pass percentage per compile:

14.4%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

Question 4 (Language: C++)

The function/method ***patternPrint*** accepts an argument *num*, an integer.
The function/method ***patternPrint*** prints *num* lines in the following pattern.

For example, *num* = 4, the pattern should be:

```
1
1 1
1 1 1
1 1 1 1
```

The function/method ***patternPrint*** compiles successfully but fails to print the desired result for some test cases due to incorrect implementation of the function/method. Your task is to fix the code so that it passes all the test cases.

Scores

Final Code Submitted

Compilation Status: Pass

```
1 // You can print the values to stdout for debugging
2 using namespace std;
3 void patternPrint(int num)
4 {
5     int print=1;
6     for(int i=0;i<num;i++)
7     {
8         for(int j=0;j<=i;j++)
9         {
10             cout<<print<<" ";
```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code:

*N represents

```

11  }
12  cout<<"\n";
13  }
14 }
15

```

Errors/Warnings

There are no errors in the candidate's code.


Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 100%

Total score

 8/8

100%

Basic(7/7)

0%

Advance(0/0)

100%

Edge(1/1)

Compilation Statistics

4

Total attempts

4

Successful

0

Compilation errors

2

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:02:13

Average time taken between two compile attempts:

00:00:33

Average test case pass percentage per compile:

56.3%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

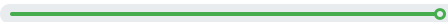
Question 5 (Language: C++)

The function/method **matrixSum** returns an integer representing the sum of elements of the input matrix. The function/method **matrixSum** accepts three arguments - *rows*, an integer representing the number of rows of the input matrix, *columns*, an integer representing the number of columns of the input matrix and *matrix*, a two-dimensional array representing the input matrix.

The function/method **matrixSum** compiles unsuccessfully due to syntactical error. Your task is to debug the program so that it passes all test cases.

Scores

Final Code Submitted	Compilation Status: Pass	Code Analysis
<pre> 1 // You can print the values to stdout for debugging 2 using namespace std; 3 int matrixSum(int rows, int columns, int **matrix) 4 { 5 int i, j, sum=0; 6 for(i=0;i<rows;i++) 7 { 8 for(j=0;j<columns;j++) 9 sum += matrix[i][j]; 10 } 11 return sum; 12 }</pre>		Average-case Time Complexity <p>Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.</p> <p>Best case code:</p> <p>*N represents</p>
		Errors/Warnings <p>There are no errors in the candidate's code.</p>
		Structural Vulnerabilities and Errors <p>There are no errors in the candidate's code.</p>

Test Case Execution	Passed TC: 100%		
Total score  10/10	100% Basic(6/6)	100% Advance(4/4)	0% Edge(0/0)

Compilation Statistics					
3	2	1	0	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors
Response time:					00:00:31
Average time taken between two compile attempts:					00:00:10
Average test case pass percentage per compile:					66.7%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

Question 6 (Language: C++)

The function/method ***allExponent*** returns a real number representing the result of exponentiation of base raised to power exponent for all input values. The function/method ***allExponent*** accepts two arguments - *baseValue*, an integer representing the base and *exponentValue*, an integer representing the exponent.

The incomplete code in the function/method ***allExponent*** works only for positive values of the exponent. You must complete the code and make it work for negative values of exponent as well.

Another function/method ***positiveExponent*** uses an efficient way for exponentiation but accepts only positive *exponent* values. You are supposed to use this function/method to complete the code in ***allExponent*** function/method.

Helper Description

The following function is used to represent a positiveExponent and is already implemented in the default code (Do not write this definition again in your code):

```
int positiveExponent(int baseValue, int exponentValue)
{
    /*It calculates the Exponent for the positive value of exponentValue

    This can be called as -

    int res = (float)positiveExponent(baseValue, exponentValue);*/
}
```

Scores

Final Code Submitted

Compilation Status: Pass

```
1 // You can print the values to stdout for debugging
2 using namespace std;
3 float allExponent(int baseValue, int exponentValue)
4 {
5     float res = 1;
6     if(exponentValue >=0)
7     {
8         res = (float)positiveExponent(baseValue, exponentValue);
9     }
10    else
11    {
12        // write your code here for negative exponentInput
13        res = 1/(float)positiveExponent(baseValue, abs(exponentValue));
14    }
15 }
16 return res;
17 }
18
19
```

Code Analysis

Average-case Time Complexity

Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.

Best case code:

*N represents

Errors/Warnings

There are no errors in the candidate's code.

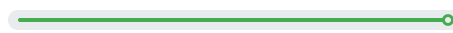
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 100%

Total score

 6/6

100%

Basic(2/2)

100%

Advance(3/3)

100%

Edge(1/1)

Compilation Statistics

4

Total attempts

4

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:01:09

Average time taken between two compile attempts:

00:00:17

Average test case pass percentage per compile:

79.2%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

Question 7 (Language: C++)

The function/method ***selectionSortArray*** performs an in-place selection sort on the given input list which will be sorted in ascending order.

The function/method ***selectionSortArray*** accepts two arguments - *len*, an integer representing the length of the input list and *arr*, a list of integers representing the input list, respectively.

The function/method ***selectionSortArray*** compiles successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

Note:

In this particular implementation of selection sort, the smallest element in the list is swapped with the element at first index, the next smallest element is swapped with the element at the next index and so on.

Scores

Final Code Submitted	Compilation Status: Pass	Code Analysis
<pre> 1 // You can print the values to stdout for debugging 2 void selectionSortArray(int len, int* arr){ 3 int x=0, y =0; 4 for(x=0; x<len; x++){ 5 int index_of_min = x; 6 for(y=x; y<len; y++){ 7 if(arr[index_of_min]>arr[y]){ 8 index_of_min = y; </pre>		Average-case Time Complexity <p>Candidate code: Complexity is reported only when the code is correct and it passes all the basic and advanced test cases.</p> <p>Best case code:</p>

```

9      }
10     }
11     int temp = arr[x];
12     arr[x] = arr[index_of_min];
13     arr[index_of_min] = temp;
14 }
15 }
```

*N represents

Errors/Warnings

There are no errors in the candidate's code.

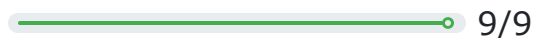
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 100%

Total score



100%

Basic(4/4)

100%

Advance(5/5)

0%

Edge(0/0)

Compilation Statistics

5

Total attempts

5

Successful

0

Compilation errors

3

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:01:10

Average time taken between two compile attempts:

00:00:14

Average test case pass percentage per compile:

46.7%

Average-case Time Complexity

Average Case Time Complexity is the order of performance of the algorithm given a random set of inputs. This complexity is measured here using the Big-O asymptotic notation. This is the complexity detected by empirically fitting a curve to the run-time for different input sizes to the given code. It has been benchmarked across problems.

Test Case Execution

There are three types of test-cases for every coding problem:

Basic: The basic test-cases demonstrate the primary logic of the problem. They include the most common and obvious cases that an average candidate would consider while coding. They do not include those cases that need extra checks to be placed in the logic.

Advanced: The advanced test-cases contain pathological input conditions that would attempt to break the codes which have incorrect/semi-correct implementations of the correct logic or incorrect/semi-correct formulation of the logic.

Edge: The edge test-cases specifically confirm whether the code runs successfully even under extreme conditions of the domain of inputs and that all possible cases are covered by the code

WriteX - Essay Writing

77 / 100

CEFR: C1

Question

Nowadays, many youngsters constantly look for job opportunities abroad and in the long term want to settle there.

In your view, what are the reasons for the same? Do you wish to settle abroad or stay back in India? Substantiate your response with reasons.

Scores

Content Score

Grammar Score

74 / 100

85 / 100

Response

In today's world, as the competition is increasing day by day people sought to seek migration as their alternative to get a better job in abroad. India is a highly populated country and due to this, the people are more and the job opportunities are less. Also, another main reason of people migrating to abroad is that the facilities such as healthcare, education, insurance, security are also better, because the government can provide these facilities to a smaller group of people easily than in a vast and populated country like India. We can see in our day to day life many examples of prominent figures in each field like sir Sundar Pichai, Satya Nadela and many more. They studied in India then moved to abroad to seek higher education and are doing great now. These were prominent figures, but we also see in our surroundings and in our relatives mainly in Punjab and Haryana, people are more excited and eager to move to Canada, Australia, and other countries. They appear for IELTS and get bands accordingly. The main reason behind this is same and other can be the availability of resources nowadays. A lot of coaching institutes have been opened in recent years, which are providing the education which is needed at good price. In my opinion. I would stay back in India because job is not only a factor affecting your life. There are many factors that are necessary to be taken care of such as your family, expenses and more. I would like you to think a bit like whom you are doing job for in abroad, your family, right. So, if they are not happy as they are left alone in India, so what's the purpose of that money. Family is the main reason, I would stay back in India, they love India and Indian culture. There are also good opportunities in India, if a person is skilled enough there is no starvation of resources and anything. The second factor is the expenses in abroad are very high. We can live luxury life in India with the same amount of money in which we can barely afford our daily expenses in abroad. I have seen many examples in my relatives who are moving back to India due to the same reason. In the end, I would stay back in India because my family's happiness is what matters most.

Error Summary

Spelling	8
White Space	1
Style	0
Grammar	37
Typographical	0

Essay Statistics

400

Total words

20

Total sentences

20

Average sentence
length

196

Total unique words

182

Total stop words

Error Details

Spelling

... people migrating to abroad is that the **facilites** such as h
ealthcare, education, insuranc...

Possible spelling mistake found

..., security are also better, because the **gouvernement** can
provide these facilites to a smal...

Possible spelling mistake found

...cause the gouvernement can provide these **facilitites** to a
smaller group of people easily tha...

Possible spelling mistake found

...rominent figures in each field like sir **Sundar** Pichai, Saty
a Nadela and many more. The...

Possible spelling mistake found

...t figures in each field like sir Sundar **Pichai**, Satya Nadela
and many more. They studi...

Possible spelling mistake found

...ach field like sir Sundar Pichai, Satya **Nadela** and many
more. They studied in India th...

Possible spelling mistake found

...d in our relatives mainly in Punjab and **Haryana**, people
are more excited and eager to m...

Possible spelling mistake found

...ehind this is same and other can be the **availability** of res
ources nowadays. A lot of coachin...

Possible spelling mistake found

White Space

...oviding the education which is needed at good price. In
my opinion. I would stay...

Possible typo: you repeated a whitespace

Grammar

In **today's** world, as the competition is increasing day by d
ay people sought to seek migration as their alternative to g
et a better job in abroad.

Possible grammar error found. Consider replacing it with
"today".

In today's **world**, as the competition is increasing day by d
ay people sought to seek migration as their alternative to g
et a better job in abroad.

Possible grammar error found. Consider inserting "is" over
here.

In today's world, as the competition is increasing day by **d**
ay people sought to seek migration as their alternative to g
et a better job in abroad.

Possible grammar error found. Consider replacing it with
"day,".

In today's world, as the competition is increasing day by d
ay people **sought** to seek migration as their alternative to g
et a better job in abroad.

Possible grammar error found. Consider replacing it with
"seek".

In today's world, as the competition is increasing day by day people sought to seek migration as their alternative to **get** a better job in abroad.

Possible grammar error found. Consider replacing it with "getting".

In today's world, as the competition is increasing day by day people sought to seek migration as their alternative to **get** a better job **in** abroad.

Possible grammar error found. Consider removing "in" from here.

India is a highly populated country and due to this, **the** people are more and the job opportunities are less.

Possible grammar error found. Consider removing "the" from here.

India is a highly populated country and due to this, the people are more and **the** job opportunities are less.

Possible grammar error found. Consider removing "the" from here.

Also, another main reason **of** people migrating to abroad is that the facilities such as healthcare, education, insurance, security are also better, because the government can provide these facilities to a smaller group of people easily than in a vast and populated country like India.

Possible grammar error found. Consider replacing it with "for".

Also, another main reason of people migrating **to** abroad is that the facilities such as healthcare, education, insurance, security are also better, because the government can provide these facilities to a smaller group of people easily than in a vast and populated country like India.

Possible grammar error found. Consider removing "to" from here.

Also, another main reason of people migrating to abroad is that **the** facilities such as healthcare, education, insurance, security are also better, because the government can provide these facilities to a smaller group of people easily than in a vast and populated country like India.

Possible grammar error found. Consider removing "the" from here.

We can see in our day to day life many examples of prominent figures in each **field** like sir Sundar Pichai, Satya Nadel a and many more.

Possible grammar error found. Consider replacing it with "field,".

They studied in **India** then moved to abroad to seek higher education and are doing great now.

Possible grammar error found. Consider replacing it with "India,".

They studied in India then moved **to** abroad to seek higher education and are doing great now.

Possible grammar error found. Consider removing "to" from here.

These were prominent figures, but we also see in our surroundings and in our **relatives** mainly in Punjab and Haryana, people are more excited and eager to move to Canada, Australia, and other countries.

Possible grammar error found. Consider replacing it with "relatives,".

The main reason behind this is **same** and other can be the availability of resources nowadays.

Possible grammar error found. Consider inserting "the" over here.

A lot of coaching institutes have been opened in recent years, which **are providing** the education which is needed at good price.

Possible grammar error found. Consider replacing it with "provide".

A lot of coaching institutes have been opened in recent years, which are providing **the** education which is needed at good price.

Possible grammar error found. Consider removing "the" from here.

A lot of coaching institutes have been opened in recent years, which are providing the education which is needed at good price.

Possible grammar error found. Consider inserting "a" over here.

I would stay back in India because **job** is not only factor affecting your life.

Possible grammar error found. Consider inserting "my" over here.

I would stay back in India because job is not **only** factor affecting your life.

Possible grammar error found. Consider inserting "the" over here.

I would like you to think a bit **like** whom you are doing job for in abroad, your family, right.

Possible grammar error found. Consider replacing it with "about".

I would like you to think a bit like whom you are doing **job** for in abroad, your family, right.

Possible grammar error found. Consider replacing it with "jobs".

I would like you to think a bit like whom you are doing job **for in** abroad, your family, right.

Possible grammar error found. Consider removing "for in" from here.

So, if they are not happy as they are left alone in India, so what's **the** purpose of that money.

Possible grammar error found. Consider inserting "is" over here.

So, if they are not happy as they are left alone in India, so what's the purpose of that **money**.

Possible grammar error found. Consider replacing it with "money?".

Family is the main reason, I would stay back in **India**, they love India and Indian culture.

Possible grammar error found. Consider replacing it with "India".

There are also good opportunities in **India**, if a person is skilled enough there is no starvation of resources and anything.

Possible grammar error found. Consider replacing it with "India".

The second factor is **the** expenses in abroad are very high.

Possible grammar error found. Consider replacing it with "that".

The second factor is the expenses **in** abroad are very high.

Possible grammar error found. Consider removing "in" from here.

We can live luxury life in India with the same amount of money **in** which we can barely afford our daily expenses in a broad.

Possible grammar error found. Consider removing "in" from here.

We can live luxury life in India with the same amount of money in which we can barely afford **our** daily expenses in a broad.

Possible grammar error found. Consider inserting "for" over here.

We can live luxury life in India with the same amount of money in which we can barely afford our daily expenses **in** a broad.

Possible grammar error found. Consider removing "in" from here.

I have seen many examples **in** my relatives who are moving back to India due to the same reason.

Possible grammar error found. Consider replacing it with "of".

I have seen many examples in my relatives who are moving back to India **due to** the same reason.

Possible grammar error found. Consider replacing it with "for".

In the end, I would stay back in India because my **family's** happiness is what matters most.

Possible grammar error found. Consider replacing it with "family".

In the end, I would stay back in India because my family's **happiness** is what matters most.

Possible grammar error found. Consider inserting "is" over here.

4 | Learning Resources

English Comprehension			
Improve your knowledge of Business English			
Improve your hold on the language by reading Shakespearan plays			
Learn about how to get better at reading			
Logical Ability			
Take a course on advanced logic			
Test your deduction skills!			
Learn about advanced deductive logic			
Quantitative Ability (Advanced)			
Watch a video on the history of algebra and its applications			
Learn about proportions and its practical usage			
Learn about calculating percentages manually			
Icon Index			
Free Tutorial	Paid Tutorial	Youtube Video	Web Source
Wikipedia	Text Tutorial	Video Tutorial	Google Playstore