

Mohit

Test ID: 450011715000131 | 9625577151 | mohit9625577@gmail.com

Test Date: April 26, 2025

Computer Science 100 /100	Logical Ability 65 /100	Computer Programming 52 /100	Quantitative Ability (Advanced) 85 /100
English Comprehension 78 /100	WriteX - Essay Writing 82 /100	Automata 94 /100	Automata Fix 0 /100
Personality Completed			

Computer Science			100 / 100
OS and Computer Architecture	DBMS	Computer Networks	
100 / 100	89 / 100	100 / 100	

Logical Ability			65 / 100
Inductive Reasoning	Deductive Reasoning	Abductive Reasoning	
63 / 100	71 / 100	62 / 100	

Computer Programming

52 / 100

Basic Programming

49 / 100

Data Structures

57 / 100

OOP and Complexity Theory

50 / 100

Quantitative Ability (Advanced)

85 / 100

Basic Mathematics

87 / 100

Advanced Mathematics

82 / 100

Applied Mathematics

85 / 100

English Comprehension

78 / 100

CEFR: **C1**

Grammar

86 / 100

Vocabulary

75 / 100

Comprehension

73 / 100

WriteX - Essay Writing

82 / 100

CEFR: **C1**

Content Score

88 / 100

Grammar Score

70 / 100

Automata

94 / 100

Programming Ability

90 / 100

Programming Practices

100 / 100

Functional Correctness

69 / 100

Automata Fix

0 / 100

Logical Error

0 / 100

Code Reuse

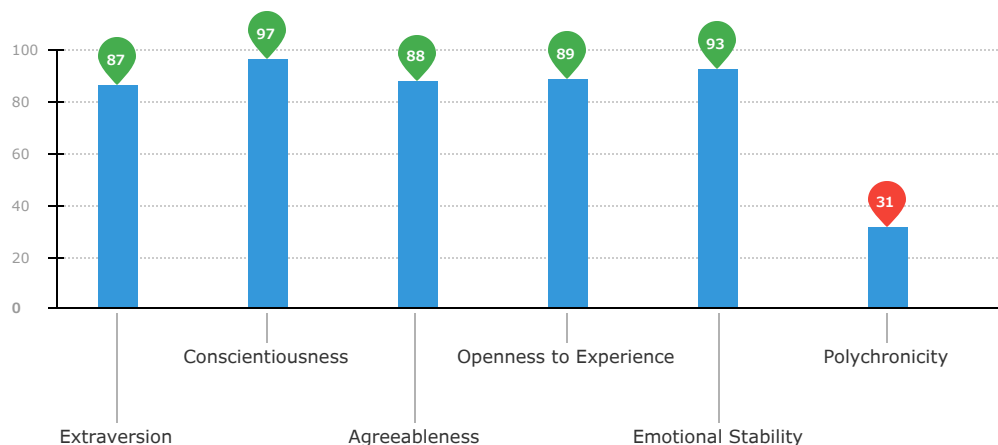
0 / 100

Syntactical Error

0 / 100

Personality

Completed

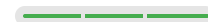


Competencies

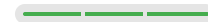
People Interaction



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

78 / 100

CEFR: **C1**

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

You have a fairly rich vocabulary and a strong command of English grammar. You are able to read and understand complex text. Having a good command of the English language is important to communicate with internal stakeholders and clients, as well as to interpret reports, articles and complex texts at work.

Logical Ability

65 / 100



Inductive Reasoning

63 / 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

71 / 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

62 / 100

Quantitative Ability (Advanced)

85 / 100

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

You have excellent quantitative ability. You are able to apply mathematical concepts to real-life problems to analyze, solve and make predictions.

Personality

Competencies



Extraversion

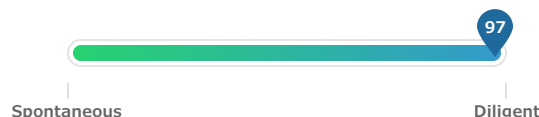


Extraversion refers to a person's inclination to prefer social interaction over spending time alone. Individuals with high levels of extraversion are perceived to be outgoing, warm and socially confident.

- 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 prefer to work on one task at a time, complete it and then move on to the next.
- You prefer orderliness and likes to concentrate on the task at hand without any distractions.
- You can find it difficult to be placed in a work environment where there is a need to multitask or where expected to engage in multiple projects simultaneously.

3 | Response

Automata



94 / 100

[Code Replay](#)

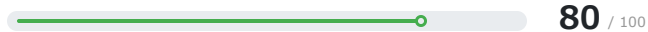
Question 1 (Language: C++)

In a science research lab, combining two nuclear chemicals produces a maximum energy that is the product of the energy of the two chemicals. The energy values of the chemicals can be negative or positive. The scientist wants to calculate the sum of the energies of the two chemicals which produces maximum energy on reaction.

Write an algorithm to find the sum of the energy of the two chemicals which produces maximum energy on reaction.

Scores

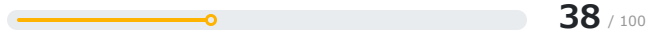
Programming Ability



80 / 100

Correct with inadvertent errors. Correct control structures and critical data dependencies incorporated. Some inadvertent errors make the code fail test cases.

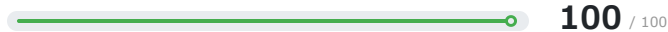
Functional Correctness



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 #include <climits>
6 using namespace std;
7
8
9 /*
10 *
11 */
12 int maxEnergy (vector<int> ener)
13 {
14     int answer = 0;
15     int maxProduct = INT_MIN;
16     for(int i=0; i< ener.size(); i++){

```

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: $O(\log N)$

*N represents number of chemicals.

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilities and Errors

There are no errors in the candidate's code.


```

17     for(int j = i + 1;j<ener.size();j++){
18         int product = ener[i] * ener[j];
19         if(product > maxProduct){
20             maxProduct = product;
21             answer = ener[i] + ener[j];
22         }
23     }
24 }
25 // Write your code here
26
27
28 return answer;
29 }
30
31 int main()
32 {
33
34     //input for ener
35     int ener_size;
36     cin >> ener_size;
37     vector<int> ener;
38     for ( int idx = 0; idx < ener_size; idx++ )
39     {
40         int temp;
41         cin >> temp;
42         ener.push_back(temp);
43     }
44
45     int result = maxEnergy(ener);
46     cout << result;
47
48
49     return 0;
50 }
51

```

Test Case Execution

Passed TC: **84.62%**

Total score



11/13

88%

Basic(7/8)

75%

Advance(3/4)

100%

Edge(1/1)

Compilation Statistics

7

Total attempts

2

Successful

5

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:26:25

Average time taken between two compile attempts:

00:03:46

Average test case pass percentage per compile:

24.18%

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++)

High similarity detected : 96%

You are given a list of integers and an integer K . Write an algorithm to find the number of elements in the list that are strictly less than K .

Scores

Programming Ability



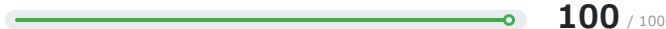
Completely correct. A correct implementation of the problem using the right control-structures and data dependencies.

Functional Correctness



Functionally correct source code. Passes all the test cases in the test suite for a given problem.

Programming Practices



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  * element, representing the vector with size of element_size.
10  * num, representing the integer to be compared(K).
11  */
12 int noOfElement (vector<int> element, int num)
13 {
14     int answer = 0;
15     for (int i = 0;i<element.size();i++){
16         if(element[i]< num){
17             answer++;
18         }
19     }
20     // Write your code here
21
22
23     return answer;
24 }
25
26 int main()
27 {
28
29     //input for element
30     int element_size;
31     cin >> element_size;
32     vector<int> element;
```

Code Analysis

Average-case Time Complexity

Candidate code: $O(N)$

Best case code: $O(N)$

*N represents number of elements in the input list

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

```

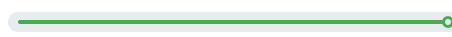
33 for ( int idx = 0; idx < element_size; idx++ )
34 {
35     int temp;
36     cin >> temp;
37     element.push_back(temp);
38 }
39 //input for num
40 int num;
41 cin >> num;
42
43
44 int result = noOfElement(element, num);
45 cout << result;
46
47
48 return 0;
49 }
50

```

Test Case Execution

Passed TC: 100%

Total score

 16/16

100%

Basic(6/6)

100%

Advance(8/8)

100%

Edge(2/2)

Compilation Statistics

3

Total attempts

3

Successful

0

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:16:38

Average time taken between two compile attempts:

00:05:33

Average test case pass percentage per compile:

100%

Similarity Detected

Every response is checked against:

- The responses of peers who took this assessment in the same event
- All candidate responses submitted in the last week
- Content currently available on the web

This response has a high degree of similarity to one of these sources, which means it is possibly not the candidate's original work:

Candidate Response

Peer's Response

```

11 */
12 int noOfElement (vector<int> element, int num)
13 {
14     int answer = 0;
15     for (int i = 0; i < element.size(); i++) {
16         if (element[i] < num) {
17             answer++;
18         }
19     }
20     // Write your code here
21
22
23     return answer;
24 }
25

```

```

11 */
12 int noOfElement (vector<int> element, int num)
13 {
14     int answer=0;
15     for(int i=0;i<element.size();i++){
16         if(element[i]<num){
17             answer++;
18         }
19     }
20
21     return answer;
22

```

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 1 (Language: C++)

The function/method **sortArray** modify the input list by sorting its elements in descending order.
The function/method **sortArray** accepts two arguments - *len*, representing the length of the list and *arr*, a list of integers representing the input list, respectively.

The function/method **sortArray** 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 sortArray(int len, int* arr)
3 {
4     int i, max, location, j, temp;
5     for(int i=0;i<len-1; i++)
6     {
7         max = arr[i];
8         location = i;
9         for(j=i+1;j<len;j++)
10        {
11            if(max<arr[j])
12            {
13                max = arr[j];
14                location = j;
15            }
16        }
17        temp=arr[i];
18        arr[i]=arr[location];
19        arr[location]=temp;
20    }
21 }
```

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.

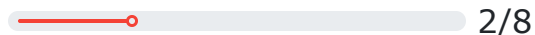
Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 25%

Total score



33%

Basic(1/3)

0%

Advance(0/4)

100%

Edge(1/1)

Compilation Statistics

1

Total attempts

1

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:09:57

Average time taken between two compile attempts:

00:09:57

Average test case pass percentage per compile:

12.5%

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++)

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    }
14    return res;
15 }
16
17
```

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.

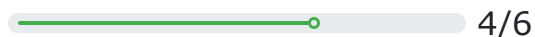
Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 66.67%

Total score



100%

Basic(2/2)

33%

Advance(1/3)

100%

Edge(1/1)

Compilation Statistics

0

Total attempts

0

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:00:21

Average time taken between two compile attempts:

00:00:00

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 3 (Language: C++)

The function/method ***replaceValues*** is modifying the input list in such a way - if the length of input list is odd, then all the elements of the input list are supposed to be replaced by 1s and in case it is even, the elements should be replaced by 0s.

For example: given the input list [0 1 2], the function will modify the input list like [1 1 1]

The function/method ***replaceValues*** accepts two arguments - *size*, an integer representing the size of the given input list and *inputList*, a list of integers representing the input list.

The function/method ***replaceValues*** compiles successfully but fails to get the desired result for some test cases due to incorrect implementation of the function. 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 replaceValues(int size, int *inputList)
3 {
4     int i, j;
5     if( size % 2 == 0 )
6     {
7         i=0;
8         while(i<size)
9         {
10             inputList[i] = 0;
11             i+=2;
12         }
13     }
14     else
15     {
16         j=0;
17         while(j<size)
18         {
19             inputList[j] = 1;
20             j+=2;
21         }
22     }
23 }
24
```

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.

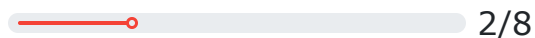
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 25%

Total score



0%

Basic(0/6)

0%

Advance(0/0)

100%

Edge(2/2)

Compilation Statistics

1

Total attempts

1

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:07:28

Average time taken between two compile attempts:

00:07:28

Average test case pass percentage per compile:

0%

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 ***calculateMatrixSum*** returns an integer representing the sum of odd elements of the given matrix whose i^{th} and j^{th} index are the same.

The function/method ***calculateMatrixSum*** accepts three arguments - *rows*, an integer representing the number of rows of the given matrix, *columns*, an integer representing the number of columns of the given matrix and *matrix*, representing a two-dimensional array of integers.

The function/method ***calculateMatrixSum*** compiles successfully but fails to return 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 using namespace std;
3 int calculateMatrixSum(int rows, int columns, int **matrix)
4 {
5     int i, j, sum=0;
6     if((rows>0) && (columns>0))
7     {
8         for(i=0;i<rows;i++)
9         {
10             sum =0;
11             for(j=0;j<columns;j++)
12             {
```

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

```

13     if(i==j)
14     {
15         if(matrix[i][j]/2!=0)
16             sum += matrix[i][j];
17     }
18 }
19 }
20 return sum;
21 }
22 else
23     return sum;
24 }
25

```

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: **12.5%**

Total score

 1/8

0%

Basic(0/4)

0%

Advance(0/3)

100%

Edge(1/1)

Compilation Statistics

0

Total attempts

0

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:00:20

Average time taken between two compile attempts:

00:00:00

Average test case pass percentage per compile:

12.5%

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 **maxReplace** prints space separated integers representing the input list after replacing all elements of the input list with the maximum element of the input list.

The function/method **maxReplace** accept two arguments - *size*, an integer representing the size of the input list and *inputList*, a list of integers representing the input list, respectively.

The function/method **maxReplace** compiles unsuccessfully due to compilation error. Your task is to debug the code so that it passes all the test cases.

Scores

Final Code Submitted

Compilation Status: Fail

```
1 // You can print the values to stdout for debugging
2 using namespace std;
3 void maxReplace(int size, int &inputList)
4 {
5     int i;
6     if(size>0)
7     {
8         int max = inputList[0];
9         for(i=0;i<size;i++)
10        {
11            if(max<inputList[i])
12            {
13                max = inputList[i];
```

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

```

14     }
15     }
16 }
17 for(i=0;i<size,i++)
18 {
19     inputList[i]=max;
20     cout<<inputList[i]<<" ";
21 }
22 }

```

In file included from main_33.cpp:7:
source_33.cpp: In function 'void maxReplace(int, int&)':
source_33.cpp:8:30: error: invalid types 'int[int]' for array subscript
int max = inputList[0];
^
source_33.cpp:11:31: error: invalid types 'int[int]' for array subscript
if(max ^
source_33.cpp:13:34: error: invalid types 'int[int]' for array subscript
max = inputList[i];
^
source_33.cpp:17:23: error: expected ';' before ')' token
for(i=0;i ^
;
source_33.cpp:19:20: error: invalid types 'int[int]' for array subscript
inputList[i]=max;
^
source_33.cpp:20:26: error: invalid types 'int[int]' for array subscript
cout< ^
main_33.cpp: In function 'int main(int, const char**)':
main_33.cpp:24:22: error: invalid conversion from 'int*' to 'int' [-fpermissive]
maxReplace(len, arr);
^~~
In file included from main_33.cpp:7:
source_33.cpp:3:32: note: initializing argument 2 of 'void maxReplace(int, int&)'
void maxReplace(int size, int &inputList)
~~~~~^~~~~~  
main\_33.cpp:24:22: error: cannot bind rvalue '(int)arr' to 'int&'  
maxReplace(len, arr);  
^~~

#### Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

## Compilation Statistics

0

Total attempts

0

Successful

0

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:01:30

Average time taken between two compile attempts:

00:00:00

Average test case pass percentage per compile:

0%

### 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: Java)

You are given predefined structure **Time** containing *hour*, *minute*, and *second* as members. A collection of functions/methods for performing some common operations on times is also available. You must make use of these functions/methods to calculate and return the difference.

The function/method **difference\_in\_times** accepts two arguments - *time1*, and *time2*, representing two times and is supposed to return an integer representing the difference in the number of seconds.

You must complete the code so that it passes all the test cases.

.

### Helper Description

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

```
public class Time
{
    public int hour;
    public int minute;
    public int second;

    public Time(int h, int m, int s)
    {
        hour = h;
        minute = m;
        second = s;
    }

    public int compareTo(Object anotherTime)
    {
        /*Return 1, if time1 is greater than time2.
        Return -1 if time1 is less than time2
        or, Return 0, if time1 is equal to time2

        This can be called as -
        * If time1 and time2 are two Time then -
        * time1.compareTo(time2) */
    }

    public void addSecond()
    {
        /* Add one second in the time;

        This can be called as -
        * If time1 is Time then -
        * time1.addSecond() */
    }
}
```



## Scores

The candidate did not make any changes in the code.

## Compilation Statistics

0

Total attempts

0

Successful

0

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:00:00

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

## Question 7 (Language: C++)

The function/method **productMatrix** accepts three arguments - *rows*, an integer representing the rows of the matrix; *columns*, an integer representing the columns of the matrix and *matrix*, a two-dimensional array of integers, respectively.

The function/method **productMatrix** return an integer representing the product of the odd elements whose  $i^{\text{th}}$  and  $j^{\text{th}}$  index are the same. Otherwise, it returns 0.

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

## Scores

### Final Code Submitted

Compilation Status: Pass

```
1 int productMatrix(int rows, int columns, int **matrix)
2 {
3     int result=0;
4     for(int i=0;i<rows;i++)
5     {
6         for(int j=0;j<columns;j++)
7         {
8             if((i==j) || (matrix[i][j]%2!=0))
9                 result *=matrix[i][j];
10        }
11    }
12    if(result<=1)
13        return 0;
14    else
15        return result;
16 }
17
```

### 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: 25%

Total score

 2/8

0%

Basic(0/5)

100%

Advance(2/2)

0%

Edge(0/1)

### Compilation Statistics

0

Total attempts

0

Successful

0

Compilation errors

1

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:00:10

Average time taken between two compile attempts:

00:00:00

Average test case pass percentage per compile:

25%

### 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

 82 / 100

CEFR: **C1**

### Question


**The use of messaging apps like WhatsApp/WeChat have bridged the communication gap between people living far apart.**

Do you agree or disagree? In your view, how have these apps transformed relationships? Substantiate your response with reasons and suitable examples.

### Scores

Content Score

Grammar Score





 88 / 100

 70 / 100

### Response

In today's world I completely agree that messaging apps like whatsapp and wechat have bridged the communication gap between people living far apart. A few decades ago, it was very difficult and expensive to stay in touch with friends and family who lived in different cities or countries. People had to rely on letters or costly international phone calls, which were often time consuming and sometimes unreliable. However with the invention of messaging apps, communication has become instant, easy and almost free. Now, whether someone is living next door or on another continent, it hardly matters, A message, a voice note, or even a video call can reach them within seconds. This has made the world feel much smaller and more connected. Messaging apps have transformed relationships in many ways. First they have made it possible for families to remain closely connected despite physical distance for example many students go abroad for higher studies. Earlier parents could only speak to them once in a while because of high

### Error Summary

|                                                                                       |             |    |
|---------------------------------------------------------------------------------------|-------------|----|
|  | Spelling    | 11 |
|  | White Space | 17 |
|  | Style       | 0  |
|  | Grammar     | 12 |

call charges. Now with apps like **whatsapp**s they can send daily updates pictures and videos and even have long conversations without worrying about cost. This regular communication helps in **maintaining** emotional closeness and reduces the feeling of homesickness. Secondly these apps have made it easier to form new relationships. Online groups based on common interests, studies or work bring together people from different parts of the world. **Friendships** and collaborations that would have been impossible earlier are now common. Another important change is **in** how special moments are shared. Birthdays, festivals and even daily joys are celebrated through group **chats** video calls and shared **photos** for instance during the **covid 19 pandemic** when people could not meet physically messaging apps became a lifeline. Families **organized** virtual **gatherings** friends played online games together and people checked **on** each other's **health** and **wellbeing** through quick **messages** apps did not just maintain existing bonds but made them even stronger.

Typographical 0

### Essay Statistics

324

Total words

17

Total sentences

19

Average sentence length

225

Total unique words

109

Total stop words

### Error Details

#### Spelling

In today's world I **completely** agree that messaging apps like whatsapp...

Possible spelling mistake found

apps like whatsapp and **wechat** have bridged the communication

Possible spelling mistake found. Consider replacing the highlighted text with: 'we chat'.

...essaging apps like whatsapp and wechat **jave** bridged the communication gap between p...

Possible spelling mistake found

...tional phone calls, which were often **time consuming** and sometimes unreliable. However wit...

This word is normally spelled with hyphen.

...made it possible for families to remain **cloasely** connected despite physical distance for...

Possible spelling mistake found

...f high call charges. Now with apps like **whatsapp**s they can send daily updates pictures and...

Possible spelling mistake found

...ost. This regular communication helps in **manitaining** emotional closeness and reduces the fee...

Possible spelling mistake found

...d shared photos for instance during the **covid 19 pandemic** when people could not meet ...

Possible spelling mistake found

...photos for instance during the covid 19 **pandamic** when people could not meet physically m...

Possible spelling mistake found

...saging apps became a lifeline. Families **oragnized** virtual gatherings friends played online...

Possible spelling mistake found

...ple checked on each other's health and **wellbeing** through quick messages apps did not jus...

Possible spelling mistake found

## White Space

In today's world I completely agree that messaging apps like whatsapp and w...

Possible typo: you repeated a whitespace

...mpletely agree that messaging apps like whatsapp and wechat have bridged the com...

Possible typo: you repeated a whitespace

...tion gap between people living far apart . A few decades ago, it was very difficul...

Don't put a space before the full stop

...A few decades ago, it was very difficult and expensive to stay in touch with fri...

Possible typo: you repeated a whitespace

...go, it was very difficult and expensive to stay in touch with friends and family...

Possible typo: you repeated a whitespace

...to stay in touch with friends and family who lived in different cities or countri...

Possible typo: you repeated a whitespace

...ters or costly international phone calls , which were often time consuming and ...

Put a space after the comma, but not before the comma

...costly international phone calls , which were often time consuming and sometime...

Possible typo: you repeated a whitespace

... international phone calls , which were often time consuming and sometimes unr...

Possible typo: you repeated a whitespace

...alls , which were often time consuming and sometimes unreliable. However with ...

Possible typo: you repeated a whitespace

...ere often time consuming and sometimes unreliable. However with the invention o...

Possible typo: you repeated a whitespace

...g apps, communication has become instant , easy and almost free. NOW, whether some...

Put a space after the comma, but not before the comma

...atters, A message, a voice note, or even a video call can reach them within secon...

Possible typo: you repeated a whitespace

...ance for example many students go abroad for higher studies. Earlier parents coul...

Possible typo: you repeated a whitespace

...ivals and even daily joys are celebrated through group chats video calls and sh...

Possible typo: you repeated a whitespace

... even daily joys are celebrated through group chats video calls and shared phot...

Possible typo: you repeated a whitespace

...nd people checked on each other's health and wellbeing through quick messages app...

Possible typo: you repeated a whitespace

## Grammar

In today's world I completely agree that messaging apps like whatsapp and wechat have bridged the communication gap between people living far apart .

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

In today's world I completely agree that messaging apps like whatsapp and wechat have bridged the communication gap between people living far apart .

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

First they have made it possible for families to remain closely connected despite physical distance for example many students go abroad for higher studies.

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

Friendships and collaborations that would have been impossible earlier are now common.

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

Another important change is in how special moments are shared.

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

Birthdays, festivals and even daily joys are celebrated through group chats video calls and shared photos for instance during the covid 19 pandemic when people could not meet physically messaging apps became a lifeline.

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

Birthdays, festivals and even daily joys are celebrated through group chats video calls and shared photos for instance during the covid 19 pandemic when people could not meet physically messaging apps became a lifeline.

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

Birthdays, festivals and even daily joys are celebrated through group chats video calls and shared photos for instance during the covid 19 pandemic when people could not meet physically messaging apps became a lifeline.

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

Families organized virtual gatherings friends played online games together and people checked on each other's health and wellbeing through quick messages apps did not just maintain existing bonds but made them even stronger.

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

Families organized virtual gatherings friends played online games together and people checked on each other's health and wellbeing through quick messages apps did not just maintain existing bonds but made them even stronger.

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

Families organized virtual gatherings friends played online games together and people checked on each other's health and wellbeing through quick messages apps did not just maintain existing bonds but made them even stronger.

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

Families organized virtual gatherings friends played online games together and people checked on each other's health and wellbeing through quick messages apps did not just maintain existing bonds but made them even stronger.

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

## 4 | Learning Resources

### English Comprehension

[Read opinions to improve your comprehension](#)



[Read research papers online](#)



[Read research papers to further enhance your skills](#)



### Logical Ability

[Practice inductive reasoning on visual data](#)



[Learn about pattern identification](#)



[Practice your Inductive Reasoning Skills!](#)



### Quantitative Ability (Advanced)

[Learn about the application of Bayes' Theorem in varied fields](#)



[Learn about math mysteries!](#)



[Learn about the Four Color theorem](#)



### Icon Index



Free Tutorial



Paid Tutorial



Youtube Video



Web Source



Wikipedia



Text Tutorial



Video Tutorial



Google Playstore