

Panya Manoj Sharma

Test ID: 432005347784195 | 7032723225 | pm1784@srmist.edu.in

Test Date: June 21, 2024

Computer Programming 41 /100	English Comprehension 69 /100	Quantitative Ability (Advanced) 51 /100	Computer Science 33 /100
Logical Ability 54 /100	WriteX - Essay Writing 81 /100	Automata 18 /100	Automata Fix 29 /100
Personality Completed			

Computer Programming			41 / 100
Basic Programming	Data Structures	OOP and Complexity Theory	
43 / 100	44 / 100	37 / 100	

English Comprehension			69 / 100	CEFR: C1
Grammar	Vocabulary	Comprehension		
73 / 100	61 / 100	74 / 100		

Quantitative Ability (Advanced)

51 / 100

Basic Mathematics

54 / 100

Advanced Mathematics

50 / 100

Applied Mathematics

48 / 100

Computer Science

33 / 100

OS and Computer Architecture

34 / 100

DBMS

38 / 100

Computer Networks

24 / 100

Logical Ability

54 / 100

Inductive Reasoning

52 / 100

Deductive Reasoning

58 / 100

Abductive Reasoning

51 / 100

WriteX - Essay Writing

81 / 100

CEFR: **C1**

Content Score

81 / 100

Grammar Score

82 / 100

Automata

18 / 100

Programming Ability

20 / 100

Programming Practices

25 / 100

Functional Correctness

13 / 100

Automata Fix

29 / 100

Logical Error

25 / 100

Code Reuse

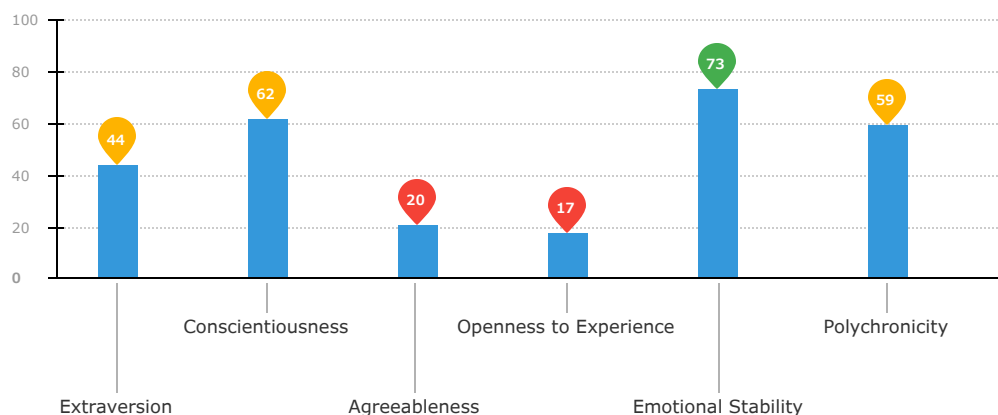
0 / 100

Syntactical Error

100 / 100

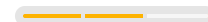
Personality

Completed

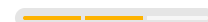


Competencies

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



69 / 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.

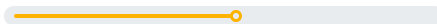
Logical Ability



54 / 100



Inductive Reasoning



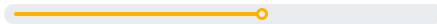
52 / 100

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

You are able to work out rules based on specific information and solve general work problems using these rules. This skill is required in data-driven research jobs where one needs to formulate new rules based on variable trends.



Deductive Reasoning



58 / 100

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

You are able to work out rules based on specific information and solve general work problems using these rules. This skill is required in data-driven research jobs where one needs to formulate new rules based on variable trends.



Abductive Reasoning



51 / 100

Quantitative Ability (Advanced)



51 / 100

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

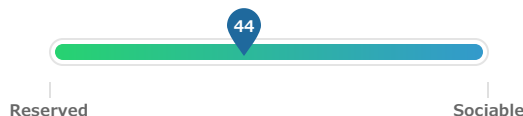
You are good at basic arithmetic. You are able to solve real-world problems that involve simple addition, subtraction, multiplication and division.

Personality

Competencies



Extraversion

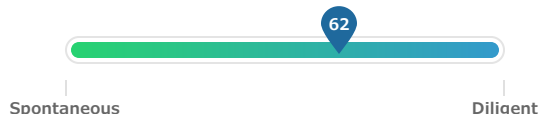


Extraversion refers to a person's inclination to prefer social interaction over spending time alone. Individuals with high levels of extraversion are perceived to be outgoing, warm and socially confident.

- You are comfortable socializing to a certain extent. You prefer small gatherings in familiar environments.
- You feel at ease interacting with your close friends but may be reserved among strangers.
- You indulge in activities involving thrill and excitement that are not too risky.
- You contemplate the consequences before expressing any opinion or taking an action.
- You take charge when the situation calls for it and you are comfortable following instructions as well.
- Your personality may be suitable for jobs demanding flexibility in terms of working well with a team as well as individually.



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 are flexible and able to adapt your work pace to the job at hand.
- You are usually spontaneous but you are likely to stick to a plan whenever necessary.
- You tend to be cautious when you deem it necessary.
- You may prefer to act according to the rules.
- You are confident in your ability to achieve goals but may need support to overcome occasional setbacks.
- You are an efficient worker and try to perform better than your peers. You are well suited for jobs allowing flexibility regarding operating procedures.



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 outspoken. You often play the role of a devil's advocate in discussions and question others' opinions and views.
- You are not gullible and are likely to carefully examine the situation before trusting something/someone.
- You may not be strongly affected by human suffering and may be perceived as indifferent.

- You are confident of your achievements and do not shy away from talking about them.
- You sometimes place self-interest above the needs of those around you. You are not willing to compromise your own views in order to accommodate the views of others.
- You are suitable for jobs that require tough objective decisions and hard negotiation.



Openness to Experience

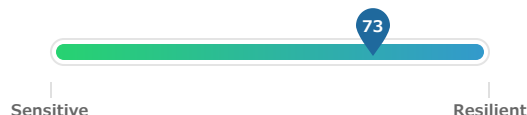


Openness to experience refers to a person's inclination to explore beyond conventional boundaries in different aspects of life. Individuals with high levels of this personality trait tend to be more curious, creative and innovative in nature.

- You may not be very open to new experiences lying outside your comfort zone and tends to prefer routine over variety.
- You may be pragmatic and is likely to be conventional in your outlook and actions and may not pursue an experimental approach to problem-solving.
- You may not have an appreciation for art.
- You do not like to express your emotions and feelings to others.
- You tend to demonstrate concrete thinking with a focus on practical solutions, as opposed to abstract ideas.
- Your personality is more suited to job roles that require logical and rational thinking.



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 neither have a strong preference nor dislike to perform multiple tasks simultaneously.
- You are open to both options - pursuing multiple tasks at the same time or working on a single project at a time.

- Whether or not you will succeed in a polychronous environment depends largely on your ability to do so.

3 | Response

WriteX - Essay Writing



81 / 100

CEFR: C1

Question

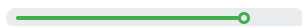
Some parents feel that sports is a distraction to their kids' studies. There are others who give due importance to sports for the holistic development of a child.

What is your view? Support your response with reasons and examples.

Scores

Content Score

Grammar Score



81 / 100



82 / 100

Response

I support the view that sports is essential for the holistic development of a child. It is a spectrum of life that every parent should encourage due to its multitude of benefits. In indulging your child in sports at an early age can foster the development of their motor skills. More than eighty to ninety percent of the parents usually consult their pediatrician for the delayed response in their child's motor skills. This would not be a concern if they do not succumb to the easier route of entertaining their child that is via electronic gadgets. Majority of the guardians tend to provide their child with an electronic gadget at a very young age to keep them from bothering them, which on one hand fulfills the agenda of keeping their antics at bay but it isn't the most optimal solution in the long run. Electronic gadgets not only limit their creative thinking but also bind them to their homes, and why would it not? when everything is available to them through at the switch of a tab. Obesity has been a major concern across various countries, especially in the first world countries. But have we ever wondered that the reason could have been more than just eating junk. Studies show that first world countries like America have the most obese people due to their unhealthy lifestyle. Majority of this problem can be avoided if people indulge in physical activities like sports. Today's generation has glorified hitting the gym since they have no idea that they can be still healthy and active by playing any sports. Sports not only helps in keeping you active and lead a healthy lifestyle but also enhances your metabolism and keeps your mental health in check. Ninety percent of the today's generation is a victim to depression which can be due to the reason that they spend majority of the day cooped up in their room with minimal interaction with the world. Sports can help you interact with the people that are like yourself and at the same time help you be active keeping any form of negative thought at bay. Therefore, I would conclude saying sports is an essential part of life.

Error Summary

	Spelling	6
	White Space	5
	Style	0
	Grammar	42
	Typographical	1

Essay Statistics

368

Total words

17

Total sentences

22

Average sentence
length

200

Total unique words

179

Total stop words

Error Details

Spelling

parent should encourage due to **it's** multitude of benefits.

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

...n sports at an early age can foster the **development** of the
ir motor skills. More than eighty...

Possible spelling mistake found

...nt of the parents usually consult their **pedeatricians** for t
he delayed response in their child...

Possible spelling mistake found

...al solution in the long run. Electronic **gadegts** not only li
mit their creative thinking ...

Possible spelling mistake found

...lge in physical activities like sports. **Today's**, generation h
as glorified hitting the g...

Possible spelling mistake found

...e thought at bays. Therefore, I would **conclude** saying s
ports is an essential part of l...

Possible spelling mistake found

White Space

...rage due to it's multitude of benefits. Indulging your chil
d in sports at an ear...

Possible typo: you repeated a whitespace

...r child that is via electronic gadgets. Majority of the guar
dians tend to provid...

Possible typo: you repeated a whitespace

...to them through at the switch of a tab. Obesity has been
a major concern across ...

Possible typo: you repeated a whitespace

...r mental health in check. Ninety percent of the today's ge
neration is a victim to ...

Possible typo: you repeated a whitespace

...g any form of negative thought at bays. Therefore, I wou
ld conclude saying spor...

Possible typo: you repeated a whitespace

Grammar

I support the view that sports **is** essential for the holistic d
evelopment of a child.

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

I support the view that sports is essential for the holistic d
evelopment of **a** child.

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

I support the view that sports is essential for the holistic d
evelopment of a **child**.

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

More than eighty to ninety percent of **the** parents usually consult their pediatricians for the delayed response in their child's motor skills.

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

More than eighty to ninety percent of the parents usually consult their pediatricians **for** the delayed response in their child's motor skills.

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

More than eighty to ninety percent of the parents usually consult their pediatricians for the delayed response **in** their child's motor skills.

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

More than eighty to ninety percent of the parents usually consult their pediatricians for the delayed response in their **child's** motor skills.

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

More than eighty to ninety percent of the parents usually consult their pediatricians for the delayed response in their child's **motor** skills.

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

This would not be a concern if they **do** not succumb to the easier of route of entertaining their child that is via electronic gadgets.

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

This would not be a concern if they do not succumb to the easier **of** route of entertaining their child that is via electronic gadgets.

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

Majority of the guardians tend to provide their **child** with a n electronic gadget at a very young age to keep them from bothering them, which on one hand fulfills the agenda of keeping their antics at bay but it isn't the most optimal solution in the long run.

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

Majority of the guardians tend to provide their child with a n electronic gadget at a very young age to keep them from bothering them, which on **one** hand fulfills the agenda of keeping their antics at bay but it isn't the most optimal solution in the long run.

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

Majority of the guardians tend to provide their child with a n electronic gadget at a very young age to keep them from bothering them, which on one **hand** fulfills the agenda of keeping their antics at bay but it isn't the most optimal solution in the long run.

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

Majority of the guardians tend to provide their child with a n electronic gadget at a very young age to keep them from bothering them, which on one hand fulfills the agenda of keeping their antics at **bay** but it isn't the most optimal solution in the long run.

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

Majority of the guardians tend to provide their child with a n electronic gadget at a very young age to keep them from bothering them, which on one hand fulfills the agenda of keeping their antics at bay but it isn't the **most** optimal solution in the long run.

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

Majority of the guardians tend to provide their child with a n electronic gadget at a very young age to keep them from bothering them, which on one hand fulfills the agenda of keeping their antics at bay but it isn't the most optimal solution in the long **run**.

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

Electronic gadegts not only limit their creative thinking but also bind them to their homes, and why would **it** not?

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

when everything is available to them **through** at the switch of a tab.

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

But have we ever wondered that the reason could have been more than just eating **junk**.

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

Majority of this problem can be avoided if people indulge in physical activities like sports.

Possible grammar error found. Consider inserting "The majority" over here.

Today's, **generation** has glorified hitting the gym since they have no idea that they can be still healthy and active by playing any sports.

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

Today's, generation has glorified hitting the gym since they have no idea that they can be **still** healthy and active by playing any sports.

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

Sports not only **helps** in keeping you active and lead a healthy lifestyle but also enhances your metabolism and keeps your mental health in check.

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

Sports not only helps **in** keeping you active and lead a healthy lifestyle but also enhances your metabolism and keeps your mental health in check.

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

Sports not only helps in **keeping** you active and lead a healthy lifestyle but also enhances your metabolism and keeps your mental health in check.

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

Sports not only helps in keeping you active and **lead** a healthy lifestyle but also enhances your metabolism and keeps your mental health in check.

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

Sports not only helps in keeping you active and lead a healthy **lifestyle** but also enhances your metabolism and keeps your mental health in check.

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

Sports not only helps in keeping you active and lead a healthy lifestyle but also **enhances** your metabolism and keeps your mental health in check.

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

Sports not only helps in keeping you active and lead a healthy lifestyle but also enhances your metabolism and **keeps** your mental health in check.

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

Ninety percent of the **today's** generation is a victim to depression which can be due to the reason that they spend majority of the day cooped up in their room with minimal interaction with the world.

Possible grammar error found. Consider removing "today's" from here.

Ninety percent of the today's generation **is** a victim to depression which can be due to the reason that they spend majority of the day cooped up in their room with minimal interaction with the world.

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

Ninety percent of the today's generation is a victim to depression which can be due to the reason that they spend majority of the day cooped up in their room with minimal interaction with the world.

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

Ninety percent of the todays generation is a **victim** to depr
ession which can be due to the reason that they spend maj
ority of the day cooped up in their room with minimal inter
action with the world.

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

Ninety percent of the todays generation is a victim **to** depr
ession which can be due to the reason that they spend maj
ority of the day cooped up in their room with minimal inter
action with the world.

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

Ninety percent of the todays generation is a victim to **depr
ession** which can be due to the reason that they spend maj
ority of the day cooped up in their room with minimal inter
action with the world.

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

Ninety percent of the todays generation is a victim to depr
ession which can be due to the reason that they spend **maj**
ority of the day cooped up in their room with minimal inter
action with the world.

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

Sports can help you interact with **the** people that are like y
ourself and at the same time help you be active keeping an
y form of negative thought at bays.

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

Sports can help you interact with the people that are like y
ourself and at the same **time** help you be active keeping an
y form of negative thought at bays.

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

Sports can help you interact with the people that are like y
ourself and at the same time help you be active keeping an
y form of negative **thought** at bays.

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

Sports can help you interact with the people that are like y
ourself and at the same time help you be active keeping an
y form of negative thought at **bays**.

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

Therefore, I would concluide saying sports **is** an essential p
art of life b

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

Therefore, I would concluide saying sports is an essential p
art of **life b**

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

Typographical

...health in check. Ninety percent of the **todays** generation
is a victim to depression wh...

Apostrophe might be missing. Did you mean "todays" or
"today's"?

Automata



18 / 100

[Code Replay](#)

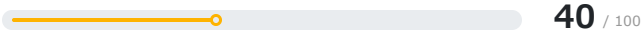
Question 1 (Language: C++)

A company is transmitting its data to another server. To secure the data against malicious activity, they plan to reverse the data before transmitting. They want to know the number of data characters that do not change position even after the data stream is reversed. The network administrator has been tasked with ensuring the smooth transmission of the data.

Write an algorithm for the network administrator to help in finding the number of data characters that do not change position even after the data stream is reversed.

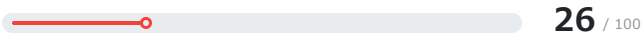
Scores

Programming Ability



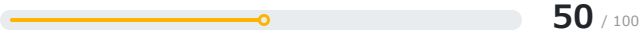
Emerging basic structure. Appropriate keywords and tokens present, showing some understanding of a part of the problem.

Functional Correctness



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



High readability, low on program structure. The source code contains redundant/improper coding constructs and a few readability and formatting issues.

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 unaffectedChar (string dataStream)
12 {
13     int answer=0;
14     string original;
15     int n = dataStream.size();
16     for(int i=0;i<n;i++){
17         original[i]=dataStream[i];
18     }
19     for(int i=n;i>0;i--){
20         dataStream[i];
21     }
22     for(int i=0;i<n/3;i++){
23         if(original[i]!=dataStream[i]){
24             answer;
25         }
26         else answer++;

```

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 size of the string.

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

Readability & Language Best Practices

Line 15: Variables are given very short name.

Performance & Correctness

Line 24: Redundant statement/expression.

```

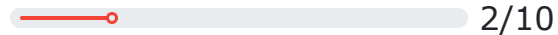
27 }
28
29 // Write your code here
30
31
32 return answer;
33 }
34
35 int main()
36 {
37
38 //input for dataStream
39 string dataStream;
40 getline(cin,dataStream);
41
42 int result = unaffectedChar(dataStream);
43 cout << result;
44
45
46 return 0;
47 }
48

```

Test Case Execution

Passed TC: **20%**

Total score



25%

Basic(1/4)

0%

Advance(0/4)

50%

Edge(1/2)

Compilation Statistics

14

Total attempts

13

Successful

1

Compilation errors

0

Sample failed

0

Timed out

9

Runtime errors

Response time:

00:16:00

Average time taken between two compile attempts:

00:01:09

Average test case pass percentage per compile:

8.57%

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++)

In a town, the houses are marked with letters in the English alphabet. A town committee wants to renovate each house. Because funds are limited, they decide to renovate only the houses marked with vowels. The committee head gives the list of houses to the members and asks them to identify the houses that will not be renovated.

Write an algorithm to help the committee members find the list of houses that will not be renovated.

Scores

Programming Ability

0 / 100

Programming ability score cannot be generated. This is because source code has syntax/ runtime errors and is unparseable.

Functional Correctness

0 / 100

Syntactically incorrect code. The source code has syntax errors in it.

Programming Practices

0 / 100

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

```
1 // Header Files
2 #include<iostream>
```

Compilation Status: Fail

Code Analysis

Average-case Time Complexity


```

3 #include<string>
4 #include<vector>
5 using namespace std;
6
7
8 /*
9 *
10 */
11 vector<string> renovateHouses (string houses)
12 {
13
14     vector<string> answer;
15     int n = houses.size();
16     string result;
17     for(int i=0;i<n;i++){
18         if(houses[i]!='a'){
19             answer.push_back(houses[i]);}
20
21     return answer;
22 }
23
24 int main()
25 {
26
27     //input for houses
28     string houses;
29     getline(cin,houses);
30
31     string result = renovateHouses(houses);
32     cout << result;
33
34
35     return 0;
36 }
37

```

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 elements in the input array

Errors/Warnings

Compiling failed with exitcode 1, compiler output:
source_747.cpp: In function 'std::vector > renovateHouses(std::__cxx11::string)':
source_747.cpp:20:39: error: no matching function for call to 'std::vector >::push_back(char&)'
answer.push_back(houses[i]);}
^
In file included from /usr/include/c++/5/vector:64:0,
from source_747.cpp:5:
/usr/include/c++/5/bits/stl_vector.h:913:7: note:
candidate: void std::vector<_Tp,
_Alloc>::push_back(const value_type&) [with _Tp =
std::__cxx11::basic_string; _Alloc = std::allocator >;
std::vector<_Tp, _Alloc>::value_type =
std::__cxx11::basic_string]
push_back(const value_type& __x)
^
/usr/include/c++/5/bits/stl_vector.h:913:7: note: no
known conversion for argument 1 from 'char' to 'const
value_type& {aka const std::__cxx11::basic_string&}'
source_747.cpp:26:1: error: a function-definition is
not allowed here before '{' token
{
^
source_747.cpp:39:1: error: expected '}' at end of
input
}
^
source_747.cpp:39:1: warning: control reaches end of
non-void function [-Wreturn-type]
}
^

Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Compilation Statistics

30

Total attempts

19

Successful

11

Compilation errors

0

Sample failed

0

Timed out

6

Runtime errors

Response time:

00:27:59

Average time taken between two compile attempts:

00:00:56

Average test case pass percentage per compile:

4.33%

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

Automata Fix



29 / 100

[Code Replay](#)

Question 1 (Language: C++)

The function/method `printCharacterPattern` accepts an integer `num`. It is supposed to print the first `num` ($0 \leq num \leq 26$) lines of the pattern as shown below.

For example, if `num = 4`, the pattern is:

```
a
ab
abc
abcd
```

The function/method compiles successfully but fails to print 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 void printCharacterPattern(int num){
4     int i, j;
5     char ch='a';
6     char print;
7     for(i=0;i<num;i++){
8         print = ch;
9         for(j=0;j<=i;j++)
10             cout<<print++;
11
12         cout<<"\n";
13     }
14 }
15
```

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

8/8

100%

Basic(3/3)

100%

Advance(4/4)

100%

Edge(1/1)

Compilation Statistics

16

Total attempts

16

Successful

0

Compilation errors

15

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:07:29

Average time taken between two compile attempts:

00:00:28

Average test case pass percentage per compile:

11.7%

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 2 (Language: C++)

The function/method ***removeElement*** prints space separated integers that remains after removing the integer at the given index from the input list.

The function/method ***removeElement*** accepts three arguments - *size*, an integer representing the size of the input list, *indexValue*, an integer representing given index and *inputList*, a list of integers representing the input list.

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

Note:

Zero-based indexing is followed to access list elements.

Scores

Final Code Submitted

Compilation Status: Pass

```
1 // You can print the values to stdout for debugging
2 using namespace std;
3 void removeElement(int size, int indexValue, int *inputList)
4 {
5     int i,j;
6     if(indexValue<size)
7     {
8         for(i=indexValue;i<size;i++)
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     if(inputList[i]!=inputList[i+1]){
11         inputList[i]=inputList[i++];
12     }
13     for(i=0;i<size;i++)
14         cout<<inputList[i]<<" ";
15 }
16 else
17 {
18     for(i=0;i<size;i++)
19         cout<<inputList[i]<<" ";
20 }
21 }
22

```

Errors/Warnings

There are no errors in the candidate's code.

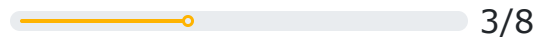
Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: **37.5%**

Total score



20%

Basic(1/5)

50%

Advance(1/2)

100%

Edge(1/1)

Compilation Statistics

6

Total attempts

6

Successful

0

Compilation errors

6

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:03:41

Average time taken between two compile attempts:

00:00:37

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 3 (Language: C++)

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 the time and is already implemented in the default code (Do not write this definition again in your code):

```
class Time
{
    int hour;

    int minute;

    int second;

    int Time :: Time_compareTo( Time* time2)

    {
```

/*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) */

}

void Time :: Time_addSecond()

{

/* Add one second in the time;

This can be called as -

* If time1 is Time then -

* time1.addSecond() */

}

Scores

Final Code Submitted

Compilation Status: Fail

```
1 // You can print the values to stdout for debugging
2 using namespace std;
3 int difference_in_times(Time *time1, Time *time2)
4 {
5     // write your code here
6 }
7
```

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

In file included from main_24.cpp:8:
source_24.cpp: In function 'int difference_in_times(Time*, Time*)':
source_24.cpp:6:1: error: no return statement in function returning non-void [-Werror=return-type]
}

^

cc1plus: some warnings being treated as errors

Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Compilation Statistics

1

Total attempts

0

Successful

1

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:00:52

Average time taken between two compile attempts:

00:00:52

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 4 (Language: C++)

The function/method **countElement** returns the number of elements in the input list *arr* which are greater than twice the input number *K*. The function/method **countElement** accepts three arguments - *size*, an integer representing the size of the input list, *numK*, an integer representing the input number *K* and *inputList*, a list of integers.

The function/method compiles unsuccessfully due to syntactical error. 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 countElement(int size, int numK, int *inputList)
4 {
5     int i,count=0;
6     for(i=0;i<size;i++)
7     {
8         if(inputList[i]>2*numK)
9             count+=1;
10    }
11    return count;
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

There are no errors in the candidate's code.

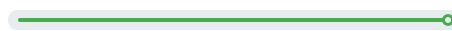
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 100%

Total score

 10/10

100%

Basic(6/6)

100%

Advance(3/3)

100%

Edge(1/1)

Compilation Statistics

3

Total attempts

1

Successful

2

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:01:09

Average time taken between two compile attempts:

00:00:23

Average test case pass percentage per compile:

33.3%

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 **arrayReverse** modify the input list by reversing its element

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

For example, if the input list *arr* is {20 30 10 40 50}, the function/method is supposed to print {50 40 10 30 20}.

The function/method **arrayReverse** 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 arrayReverse(int len, int* arr)
3 {
4     int i, temp, originalLen=len;
5     for(i=0;i<=originalLen/2;i++)
6     {
7         temp = arr[len-1];
8         arr[len-1] = arr[i];
9         arr[i] = temp;
10        len -= 1;
11    }
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

There are no errors in the candidate's code.

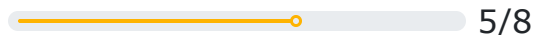
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: **62.5%**

Total score



67%

Basic(4/6)

50%

Advance(1/2)

0%

Edge(0/0)

Compilation Statistics

8

Total attempts

8

Successful

0

Compilation errors

7

Sample failed

0

Timed out

1

Runtime errors

Response time:

00:02:13

Average time taken between two compile attempts:

00:00:17

Average test case pass percentage per compile:

8.9%

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 **countDigits** return an integer representing the remainder when the given number is divided by the number of digits in it.

The function/method **countDigits** accepts an argument - *num*, an integer representing the given number.

The function/method **countDigits** compiles successfully but fails to print 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 countDigits(int num)
4 {
5     int count =0;
6     while(num!=0){
7         num=num/10;
8
9         count++;
10    }
11    return (num%count);
12 }
13
14
```

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.

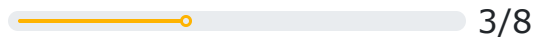
Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: 37.5%

Total score



33%

Basic(2/6)

50%

Advance(1/2)

0%

Edge(0/0)

Compilation Statistics

3

Total attempts

3

Successful

0

Compilation errors

3

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:02:14

Average time taken between two compile attempts:

00:00:45

Average test case pass percentage per compile:

8.3%

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 7 (Language: C++)

The function/method **findMaxElement** return an integer representing the largest element in the given two input lists. The function/method **findMaxElement** accepts four arguments - *len1*, an integer representing the length of the first list, *arr1*, a list of integers representing the first input list, *len2*, an integer representing the length of the second input list and *arr2*, a list of integers representing the second input list, respectively.

Another function/method **sortArray** accepts two arguments - *len*, an integer representing the length of the list and *arr*, a list of integers, respectively and return a list sorted ascending order.

Your task is to use the function/method **sortArray** to complete the code in **findMaxElement** 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 int* sortArray(int len, int* arr)
4 {
5     int i=0,j=0,temp=0;
6     for(i=0;i<len;i++)
7     {
8         for(j=i+1;j<len;j++)
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  if(arr[i]>arr[j])
11  {
12      temp = arr[i];
13      arr[i] = arr[j];
14      arr[j] = temp;
15  }
16  }
17  }
18  return arr;
19 }
20
21 int findMaxElement(int len1, int* arr1, int len2, int* arr2)
22 {
23
24  // write your code here
25 }
26

```

Errors/Warnings

In file included from main_18.cpp:5:
source_18.cpp: In function 'int findMaxElement(int, int*, int, int*)':
source_18.cpp:25:1: error: no return statement in function returning non-void [-Werror=return-type]
}
^
cc1plus: some warnings being treated as errors

Structural Vulnerabilities and Errors

There are no errors in the candidate's code.

Compilation Statistics

2

Total attempts

0

Successful

2

Compilation errors

0

Sample failed

0

Timed out

0

Runtime errors

Response time:

00:01:14

Average time taken between two compile attempts:

00:00:37

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

4 | Learning Resources

English Comprehension			
Read the latest articles by The Economist			
Read novels to enhance your comprehension skills			
Read opinions to improve your comprehension			
Logical Ability			
Learn about the fallacies in deductive reasoning			
Learn about validity of arguments			
Practice Sherlock Holmes' puzzles and develop your deductive logic			
Quantitative Ability (Advanced)			
Learn about percentages			
Learn about simple and compound interests			
Watch a video on time, speed and distance			
Icon Index			
Free Tutorial	Paid Tutorial	Youtube Video	Web Source
Wikipedia	Text Tutorial	Video Tutorial	Google Playstore