Sanyog Dani

Test ID: 432006446986932 | 9301296145 | sd6513@srmist.edu.in

Test Date: August 28, 2024

Computer Science	Logical Ability	Computer Programming	Quantitative Ability (Advanced)
73 /100	53 /100	45 /100	44 /100

English Comprehension	WriteX - Essay Writing	Automata Fix	Automata Pro
68 /100	69 /100	29 /100	5 /100

Personality

Completed

Computer Science

73 / 100

OS and Computer Architecture	DBMS	Computer Networks
89 / 100	73 / 100	30 / 100

Logical Ability

53 / 100

Inductive Reasoning	Deductive Reasoning	Abductive Reasoning
55 / 100	52 / 100	53 / 100

Computer Programming

45 / 100

Basic Programming	Data Structures	OOP and Complexity Theory
43 / 100	45 / 100	48 / 100

Quantitative Ability (Advanced)

44 / 100

Basic Mathematics	Advanced Mathematics	Applied Mathematics
45 / 100	40 / 100	46 / 100

English Comprehension

68 / 100 | CEFR: **C1**

Grammar	Vocabulary	Comprehension
70 / 100	67 / 100	68 / 100

WriteX - Essay Writing

69 / 100 | CEFR: **B1**

Content Score	Grammar Score
67 / 100	75 / 100

Automata Fix

29 / 100

Logical Error	Code Reuse	Syntactical Error
50 / 100	0 / 100	0 / 100

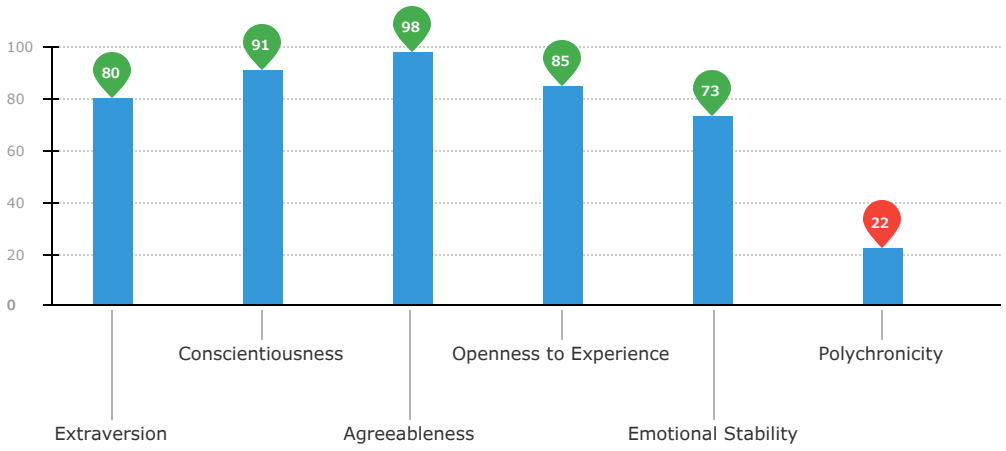
Automata Pro

5 / 100

Programming Ability	Programming Practices	Functional Correctness
10 / 100	0 / 100	11 / 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

68 / 100 | CEFR: **C1**

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

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

Logical Ability

53 / 100

Inductive Reasoning

55 / 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

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.

Abductive Reasoning

53 / 100

Quantitative Ability (Advanced)

44 / 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

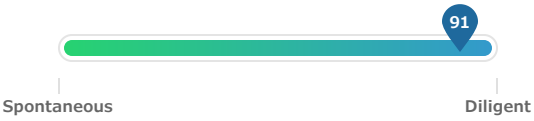
Competencies

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 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.

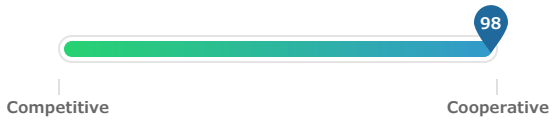
Conscientiousness



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.

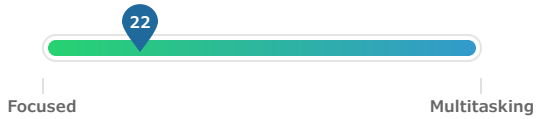
- You tend to be curious in nature and is generally open to trying new things outside your comfort zone.
- You may have a different approach to solving conventional problems and tend to experiment with those solutions.
- You are creative and tends to appreciate different forms of art.
- You are likely 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

WriteX - Essay Writing

69 / 100 | CEFR: **B1**

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
67 / 100	75 / 100

Response

I feel sports is important for the holistic development of a child and it does not distract kid's studies , study shows that playing sports simulates brain with adrenaline and help body and mind to relax and become calm . There are many advantages of playing spor ts it develops kids competitive mindset , motivates to do hard work and is very importa nt for physical health. As day by day technology is increasing and outdoor activities are decreasing , physical health is declining among yougstures resulting in irregular heart r ates and various types of other issues. It helps in overall body development of a child p roperly and helps them to become strong physically and mentally. Sports is not only ab out physical things it is also a mental game , kids develop a critical thinking technique and lear to take decisions on time , which helps in development of a kids not only physi cally but also mentally. Kids are becoming weak day by day due to not enough outdoor activities in which sports plays a major role , playing sports regularly helps tp regain st rength and stamina. Sports help kids to socialize and give them a chance to explore th e surroundings and make new friends. There is physics involved in every sports like pro jectile motion , time relativity and other topics or aspects which a kid can understand a nd practice it daily in form of sports , sports helps kids to develop critical thinking skills and brainstorm on various aspects on to how to be better at a game , learning physics to be good at a game is important and you have to have a strong and fast thinking cap abilities and quick thinking is important in any sports. There is a large scope in sports a head if a child is good at it and inrested in pursuing it , playing sports at an early age h elps a kids to determine his interest in different fields and helps to gain knowledge prac tically. For example , students are taught about projectile motion in physics through co pies and digitally , but any kids who is playing sports aldready know about projectile m otion , let it be football or cricket which will help them to understand concept easily. S o, i personally think and preffer that kids should play sports along with their studies at it helps in overall dvelopment of kids body and personality .

Error Summary

Spelling	7
White Space	15
Style	0
Grammar	36
Typographical	0

Essay Statistics

398	11	36	191	158
Total words	Total sentences	Average sentence length	Total unique words	Total stop words

Error Details

Spelling

...ng , physical health is declining among yougstures result ing in irregular heart rates and ...	Possible spelling mistake found
a critical thinking technique and lear to take decisions on	Possible spelling mistake found. Consider replacing the highlighted text with: 'learn'.
...r role , playing sports regularly helps tp regain strength and stamina. Sports hel...	Possible spelling mistake found
...orts ahead if a child is good at it and inrested in pursuing it , playing sports at an e...	Possible spelling mistake found
...ly , but any kids who is playing sports aldready know ab out projectile motion , let it b...	Possible spelling mistake found
...ept easily. So, i personally think and preffer that kids sho uld play sports along with...	Possible spelling mistake found
...th their studies at it helps in overall dvelopment of kids b ody and personality .	Possible spelling mistake found

White Space

...d and it does not distract kid's studies , study shows that playing sports simulat...	Put a space after the comma, but not before the comma
...p body and mind to relax and become calm . There are many advantages of playing sp...	Don't put a space before the full stop
...rts it develops kids competitive mindset , motivates to d o hard work and is very i...	Put a space after the comma, but not before the comma
...ng and outdoor activities are decreasing , physical health is declining among youg...	Put a space after the comma, but not before the comma
...physical things it is also a mental game , kids develop a critical thinking techni...	Put a space after the comma, but not before the comma
...nique and lear to take decisions on time , which helps in development of a kids no...	Put a space after the comma, but not before the comma
...ities in which sports plays a major role , playing sports r egularly helps tp regai...	Put a space after the comma, but not before the comma
...d in every sports like projectile motion , time relativity a nd other topics or asp...	Put a space after the comma, but not before the comma
... and practice it daily in form of sports , sports helps kids to develop critical t...	Put a space after the comma, but not before the comma

...aspects on to how to be better at a game , learning phys ics to be good at a game i...	Put a space after the comma, but not before the comma
...s good at it and inrested in pursuing it , playing sports at an early age helps a ...	Put a space after the comma, but not before the comma
...gain knowledge practically. For example , students are ta ught about projectile mo...	Put a space after the comma, but not before the comma
... in physics through copies and digitally , but any kids wh o is playing sports aldr...	Put a space after the comma, but not before the comma
...ts aldready know about projectile motion , let it be footb all or cricket which wil...	Put a space after the comma, but not before the comma
... dvelopment of kids body and personality .	Don't put a space before the full stop

Grammar

I feel sports is important for the holistic development of a child and it does not distract kid's studies , study shows th at playing sports simulates brain with adrenaline and help body and mind to relax and become calm .	Possible grammar error found. Consider replacing it with "are".
I feel sports is important for the holistic development of a child and it does not distract kid's studies , study shows th at playing sports simulates brain with adrenaline and help body and mind to relax and become calm .	Possible grammar error found. Consider replacing it with "studies.".
I feel sports is important for the holistic development of a child and it does not distract kid's studies , study shows th at playing sports simulates brain with adrenaline and help body and mind to relax and become calm .	Possible grammar error found. Consider inserting "the" over here.
I feel sports is important for the holistic development of a child and it does not distract kid's studies , study shows th at playing sports simulates brain with adrenaline and help body and mind to relax and become calm .	Possible grammar error found. Consider replacing it with "helps".
I feel sports is important for the holistic development of a child and it does not distract kid's studies , study shows th at playing sports simulates brain with adrenaline and help body and mind to relax and become calm .	Possible grammar error found. Consider inserting "the" over here.
There are many advantages of playing sports it develops ki ds competitive mindset , motivates to do hard work and is very important for physical health.	Possible grammar error found. Consider replacing it with "to".
There are many advantages of playing sports it develops ki ds competitive mindset , motivates to do hard work and is very important for physical health.	Possible grammar error found. Consider replacing it with "sports.".
There are many advantages of playing sports it develops ki ds competitive mindset , motivates to do hard work and is very important for physical health.	Possible grammar error found. Consider inserting "us" over here.
As day by day technology is increasing and outdoor activiti es are decreasing , physical health is declining among youg stures resulting in irregular heart rates and various types o f other issues.	Possible grammar error found. Consider removing "As day" from here.

As day by day technology is increasing and outdoor activiti es are decreasing , physical health is declining among youg stures resulting in irregular heart rates and various types o f other issues.	Possible grammar error found. Consider replacing it with "day,".
It helps in overall body development of a child properly an d helps them to become strong physically and mentally.	Possible grammar error found. Consider inserting "the" over here.
It helps in overall body development of a child properly an d helps them to become strong physically and mentally.	Possible grammar error found. Consider removing "properly" from here.
Sports is not only about physical things it is also a mental game , kids develop a critical thinking technique and lear t o take decisions on time , which helps in development of a kids not only physically but also mentally.	Possible grammar error found. Consider replacing it with "are".
Sports is not only about physical things it is also a mental game , kids develop a critical thinking technique and lear t o take decisions on time , which helps in development of a kids not only physically but also mentally.	Possible grammar error found. Consider replacing it with "game.".
Sports is not only about physical things it is also a mental game , kids develop a critical thinking technique and lear t o take decisions on time , which helps in development of a kids not only physically but also mentally.	Possible grammar error found. Consider replacing it with "make".
Sports is not only about physical things it is also a mental game , kids develop a critical thinking technique and lear t o take decisions on time , which helps in development of a kids not only physically but also mentally.	Possible grammar error found. Consider inserting "the" over here.
Sports is not only about physical things it is also a mental game , kids develop a critical thinking technique and lear t o take decisions on time , which helps in development of a kids not only physically but also mentally.	Possible grammar error found. Consider removing "a" from here.
Kids are becoming weak day by day due to not enough out door activities in which sports plays a major role , playing sports regularly helps tp regain strength and stamina.	Possible grammar error found. Consider replacing it with "play".
Kids are becoming weak day by day due to not enough out door activities in which sports plays a major role , playing sports regularly helps tp regain strength and stamina.	Possible grammar error found. Consider replacing it with "role.".
Sports help kids to socialize and give them a chance to exp lore the surroundings and make new friends.	Possible grammar error found. Consider replacing it with "their".
There is physics involved in every sports like projectile mot ion , time relativity and other topics or aspects which a kid can understand and practice it daily in form of sports , spo rts helps kids to develop critical thinking skills and brainsto rm on various aspects on to how to be better at a game , l earning physics to be good at a game is important and you have to have a strong and fast thinking capabilities and qui ck thinking is important in any sports.	Possible grammar error found. Consider replacing it with "sport,".

There is physics involved in every sports like projectile mot ion , time relativity and other topics or aspects which a kid can understand and practice it daily in form of sports , spo rts helps kids to develop critical thinking skills and brainsto rm on various aspects on to how to be better at a game , l earning physics to be good at a game is important and you have to have a strong and fast thinking capabilities and qui ck thinking is important in any sports.	Possible grammar error found. Consider removing "it" from here.
There is physics involved in every sports like projectile mot ion , time relativity and other topics or aspects which a kid can understand and practice it daily in form of sports , spo rts helps kids to develop critical thinking skills and brainsto rm on various aspects on to how to be better at a game , l earning physics to be good at a game is important and you have to have a strong and fast thinking capabilities and qui ck thinking is important in any sports.	Possible grammar error found. Consider inserting "the" over here.
There is physics involved in every sports like projectile mot ion , time relativity and other topics or aspects which a kid can understand and practice it daily in form of sports , spo rts helps kids to develop critical thinking skills and brainsto rm on various aspects on to how to be better at a game , l earning physics to be good at a game is important and you have to have a strong and fast thinking capabilities and qui ck thinking is important in any sports.	Possible grammar error found. Consider replacing it with "sports.".
There is physics involved in every sports like projectile mot ion , time relativity and other topics or aspects which a kid can understand and practice it daily in form of sports , spo rts helps kids to develop critical thinking skills and brainsto rm on various aspects on to how to be better at a game , l earning physics to be good at a game is important and you have to have a strong and fast thinking capabilities and qui ck thinking is important in any sports.	Possible grammar error found. Consider replacing it with "of".
There is a large scope in sports ahead if a child is good at i t and inrested in pursuing it , playing sports at an early ag e helps a kids to determine his interest in different fields a nd helps to gain knowledge practically.	Possible grammar error found. Consider replacing it with "for".
There is a large scope in sports ahead if a child is good at i t and inrested in pursuing it , playing sports at an early ag e helps a kids to determine his interest in different fields a nd helps to gain knowledge practically.	Possible grammar error found. Consider replacing it with "kid".
For example , students are taught about projectile motion i n physics through copies and digitally , but any kids who is playing sports aldready know about projectile motion , let i t be football or cricket which will help them to understand concept easily.	Possible grammar error found. Consider replacing it with "kid".
For example , students are taught about projectile motion i n physics through copies and digitally , but any kids who is playing sports aldready know about projectile motion , let i t be football or cricket which will help them to understand concept easily.	Possible grammar error found. Consider replacing it with "plays".

For example , students are taught about projectile motion i n physics through copies and digitally , but any kids who is playing sports aldready know about projectile motion , let i t be football or cricket which will help them to understand concept easily.	Possible grammar error found. Consider replacing it with "knows".
For example , students are taught about projectile motion i n physics through copies and digitally , but any kids who is playing sports aldready know about projectile motion , let i t be football or cricket which will help them to understand concept easily.	Possible grammar error found. Consider replacing it with "is".
For example , students are taught about projectile motion i n physics through copies and digitally , but any kids who is playing sports aldready know about projectile motion , let i t be football or cricket which will help them to understand concept easily.	Possible grammar error found. Consider replacing it with "cricket,".
For example , students are taught about projectile motion i n physics through copies and digitally , but any kids who is playing sports aldready know about projectile motion , let i t be football or cricket which will help them to understand concept easily.	Possible grammar error found. Consider inserting "the" over here.
So, i personally think and preffer that kids should play spor ts along with their studies at it helps in overall dvelopment of kids body and personality .	Possible grammar error found. Consider replacing it with "because".
So, i personally think and preffer that kids should play spor ts along with their studies at it helps in overall dvelopment of kids body and personality .	Possible grammar error found. Consider inserting "the" over here.
So, i personally think and preffer that kids should play spor ts along with their studies at it helps in overall dvelopment of kids body and personality .	Possible grammar error found. Consider replacing it with "kids'bodies".

Automata Pro

5 / 100 Code Replay

Question 1 (Language: C++)

A group of eight houses, represented as cells, is arranged in a straight line. Each day every cell competes with its adjacent cells. An integer value of 1 represents an active cell and value of 0 represents an inactive cell. If both the adjacent cells are either active or inactive, the cell becomes inactive the next day; otherwise it becomes active on the next day. The two cells on the ends have a single adjacent cell, so the other adjacent cell can be assumed to be always inactive. Even after updating the cell state, its previous state is considered for updating the state of other cells. The cell information of all cells should be updated simultaneously.

Write an algorithm to output the state of the cells after the given number of days.

Scores

Programming Ability

0 / 100

NA

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.

Functional Correctness

0 / 100

NA

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 vector<int> stateOfCells (vector<int> cell, int days)
12 {
13    vector<int>  answer;
14    // Write your code here
15
16
17    return answer;
18 }
19
20 int main()
21 {
22
23    //input for cell
24    int cell_size;
25    cin >> cell_size;
26    vector<int> cell;
27    for ( int idx = 0; idx < cell_size; idx++ )
28    {
29       int temp;
30       cin >> temp;
31       cell.push_back(temp);
```

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(1)

*N represents number of days

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

```
32  }
33  //input for days
34  int days;
35  cin >> days;
36
37
38  //output
39  vector<int> result = stateOfCells(cell, days);
40  for ( int idx = 0; idx < result.size() - 1; idx++ )
41  {
42     cout << result[idx] << " ";
43  }
44  cout << result[result.size() - 1];
45
46  return 0;
47 }
48
```

Test Case Execution

Passed TC: **0%**

Total score

0/14

0%	0%	0%
Basic(**0**/8)	Advance(**0**/4)	Edge(**0**/2)

Compilation Statistics

4	2	2	0	0	2
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:06:52**
Average time taken between two compile attempts:	**00:01:43**
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 2 (Language: C++)

At a nuclear power plant, electricity is generated by a fission reactor, which uses a radioactive material in a liquid state. The reactor holds V gallons of radioactive material. The plant has N vials of radioactive liquid, each with a certain mass and a certain volume. A certain amount of electricity is generated when the liquid is poured into the reactor. Now the plant's scientists want to generate more electricity. In generating more electricity, they missed one thing: The combined mass of the radioactive liquids inside the reactor must not exceed a certain critical mass M, or else the reaction could go out of its projected trajectory and require an additional cost to stop the reaction.

Write an algorithm for the scientists to generate the maximum amount of electricity from the reactor without incurring an additional cost.

Scores

Programming Ability

20 / 100

Code seems to be unrelated to the given problem.

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.

Functional Correctness

22 / 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.

Final Code Submitted

Compilation Status: Pass

```
1
2  // Header Files
3  #include<iostream>
4  #include<string>
5  #include<vector>
6  using namespace std;
7
8
9  /*
10  * The function maxElectricity takes reactorCapacity, criticalMass, to
   talVial, volume, mass, electricity as its argument, where:
11 reactorCapacity: represents the capacity of the reactor.
12 criticalMass: represents the critical mass of the reactor.
13 totalVial: represents the total number of vials.
14 volume: represents the volume of radioactive liquid which is presen
   t in each vial.
15 mass: represents the mass of liquid in each vial.
16 electricity: represents the elecetricity generated by the radioactive l
   iquid of each vial.
17  */
18 void maxElectricity (int reactorCapacity, int criticalMass, int totalVia
   l, vector<int> volume, vector<int> mass, vector<int> electricity)
19 {
20    // Write your code here
21    /*int ele1 = 0;
22    int m1= 0;
23       for(int i=0;i< totalVial;i++){
24          while(m1<criticalMass){
25             m1 += mass[i];
26             ele1 += electricity[i];
27
```

Code Analysis

Average-case Time Complexity

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

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

*N represents number of vials

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

```
28          }
29      }*/
30    int try1 = 1300;
31    cout << try1;
32
33
34 }
35
36 int main()
37 {
38    //input for reactorCapacity
39    int reactorCapacity;
40    cin >> reactorCapacity;
41
42    //input for criticalMass
43    int criticalMass;
44    cin >> criticalMass;
45
46    //input for totalVial
47    int totalVial;
48    cin >> totalVial;
49
50
51    //input for volume
52    int volume_size;
53    cin >> volume_size;
54    vector<int> volume;
55    for ( int idx = 0; idx < volume_size; idx++ )
56    {
57       int temp;
58       cin >> temp;
59       volume.push_back(temp);
60    }
61
62    //input for mass
63    int mass_size;
64    cin >> mass_size;
65    vector<int> mass;
66    for ( int idx = 0; idx < mass_size; idx++ )
67    {
68       int temp;
69       cin >> temp;
70       mass.push_back(temp);
71    }
72
73    //input for electricity
74    int electricity_size;
75    cin >> electricity_size;
76    vector<int> electricity;
77    for ( int idx = 0; idx < electricity_size; idx++ )
```

```
78  {
79      int temp;
80      cin >> temp;
81      electricity.push_back(temp);
82  }
83
84
85  maxElectricity(reactorCapacity, criticalMass, totalVial, volume, mass, electricity);
86
87  return 0;
88 }
```

Test Case Execution

Passed TC: **4.55%**

Total score

1/22

9%	0%	0%
Basic(**1**/11)	Advance(**0**/8)	Edge(**0**/3)

Compilation Statistics

22	11	11	0	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:37:27**
Average time taken between two compile attempts:	**00:01:42**
Average test case pass percentage per compile:	**1.65%**

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 ***descendingSortArray*** performs an in-place sort on the given input list which will be sorted in descending order.
The function/method ***descendingSortArray*** accepts two arguments - *len,* an integer representing the length of the input list and *arr*, a list of integers representing the input list, respectively.
5
The function/method ***descendingSortArray*** compiles successfully but fails to get the desired result for some test cases due to logical errors. Your task is to fix the code so that it passes all the test cases.

Scores

Final Code Submitted **Compilation Status: Pass**

```
1  // You can print the values to stdout for debugging
2  void descendingSortArray(int len, int* arr)
3  {
4     int small, pos, i, j, temp;
5     for(i=0; i<=len-1;i++){
6        for(j=i; j<len;j++){
7           temp = 0;
8           if(arr[i]<arr[j]){
9              temp=arr[i];
10             arr[i]=arr[j];
               arr[j]=temp;
```

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

```
11
12      }
13     }
14   }
15 }
16
```

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: **100%**

Total score

10/10

100%	100%	0%
Basic(**6**/6)	Advance(**4**/4)	Edge(**0**/0)

Compilation Statistics

3	3	0	1	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:01:55**
Average time taken between two compile attempts:	**00:00:38**
Average test case pass percentage per compile:	**70%**

Average-case Time Complexity

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

Test Case Execution

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

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

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

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

Question 2 (Language: C)

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

remember her birthday.

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

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

Scores

Final Code Submitted **Compilation Status: Pass**

```
1 // You can print the values to stdout for debugging
2 int checkBirthDay(char* month, int day)
3 {
4   if((strcmp(month,"July")) && (day==5))
5     return 1;
6   else
7     return 0;
8 }
```

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 6/9

67%	67%	0%
Basic(**4**/6)	Advance(**2**/3)	Edge(**0**/0)

Compilation Statistics

7	6	1	4	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:04:32**
Average time taken between two compile attempts:	**00:00:39**
Average test case pass percentage per compile:	**25.4%**

Average-case Time Complexity

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

Test Case Execution

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

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

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

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

Question 3 (Language: C)

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

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

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

.

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

```
struct point;

typedef struct point

{

 int X;

 int Y;

}Point;

double Point_calculateDistance(Point *point1, Point *point2)

{

   /* Return the distance between  point1 and point2;
    This can be called as -

    *  If P1 and P2 are two points then -

    *  Point_calculateDistance(P1, P2);*/

}
```

Scores

Final Code Submitted — Compilation Status: Fail

```
1 // You can print the values to stdout for debugging
2 int isTriangle(Point *P1, Point *P2, Point *P3)
3 {
4    // write your code here
5
6 }
7
8
```

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_25.c:5:
source_25.c: In function 'isTriangle':
source_25.c:6:1: error: control reaches end of non-void function [-Werror=return-type]
}
^
cc1: some warnings being treated as errors
```

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Compilation Statistics

1	0	1	0	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:01:13**
Average time taken between two compile attempts:	**00:01:13**
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 ***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  float allExponent(int baseValue, int exponentValue)
3  {
4    float res =1;
5    if(exponentValue >=0)
6    {
7      res = (float)positiveExponent(baseValue, exponentValue);
8    }
9    else
10   {
11     // write your code here for negative value of exponentValue
12
13   }
14   return res;
15 }
16
```

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

4/6

100%	33%	100%
Basic(**2**/2)	Advance(**1**/3)	Edge(**1**/1)

Compilation Statistics

3	2	1	2	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:02:33**
Average time taken between two compile attempts:	**00:00:51**
Average test case pass percentage per compile:	**11.1%**

Average-case Time Complexity

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

Test Case Execution

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

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

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

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

Question 5 (Language: C)

The function/method ***selectionSortArray*** performs an in-place selection sort on the given input list which will be sorted in ascending order.
The function/method ***selectionSortArray*** accepts two arguments - *len,* an integer representing the length of the input list and *arr*, a list of integers representing the input list, respectively.

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

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

Scores

Final Code Submitted — Compilation Status: Pass

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

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 9/9

100%	100%	0%
Basic(**4**/4)	Advance(**5**/5)	Edge(**0**/0)

Compilation Statistics

3	3	0	2	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:01:37**
Average time taken between two compile attempts:	**00:00:32**
Average test case pass percentage per compile:	**40.7%**

Average-case Time Complexity

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

Test Case Execution

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

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

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

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

Question 6 (Language: C)

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

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

Scores

Final Code Submitted **Compilation Status: Fail**

```
1  // You can print the values to stdout for debugging
2  int matrixSum(int rows, int columns, int **matrix)
3  {
4     int i, j, sum=0;
5     for(i=0;i<rows;i++)
6     {
7       for(j=0;j<columns;j++)
8         sum + = matrix(i)(j);
9     }
10    return sum;
11 }
```

Code Analysis

Average-case Time Complexity

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

Best case code:

*N represents

Errors/Warnings

In file included from main_32.c:4:

```
source_32.c: In function 'matrixSum':
source_32.c:8:15: error: expected expression before
'=' token
sum + = matrix(i)(j);
^
source_32.c:8:17: error: called object 'matrix' is not a
function or function pointer
sum + = matrix(i)(j);
^~~~~~
source_32.c:2:44: note: declared here
int matrixSum(int rows, int columns, int **matrix)
~~~~~~~^~~~~~
```

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Compilation Statistics

1	0	1	0	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:00:53**
Average time taken between two compile attempts:	**00:00:53**
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 7 (Language: C)

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

For example, *num* = 4, the pattern should be:
1
1 1
1 1 1
1 1 1 1

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

Scores

Final Code Submitted — Compilation Status: Fail

```
1  // You can print the values to stdout for debugging
2  void patternPrint(int num)
3  {
4     int print=1,i,j;
5     for(i=0;i<num;i++)
6     {
7        for(j=0;j<=i;j++);
8        {
9           for(int k=0; k<=j;k++){
10             printf("%d ",print);
11          }
12       printf("\n");
13    }
14 }
15
16
```

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

main_35.c: In function 'patternPrint':
main_35.c:16:1: error: expected declaration or statement at end of input
}
^

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Compilation Statistics

14	12	2	12	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:05:47**
Average time taken between two compile attempts:	**00:00:25**
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

4 | Learning Resources

English Comprehension

Read opinions to improve your comprehension	
Improve your vocabulary of terms used in business	
Improve your knowledge of Business English	

Logical Ability

Test your application of inductive logic!	
Play Sudoku and develop your skills of deduction!	
Learn about the art of deduction	

Quantitative Ability (Advanced)

Watch a video on time, speed and distance	
Learn about percentages	
Learn about simple and compount interests	

Icon Index