Preetam Naik

Test ID: 450013359000113 | ☎ 9676290504 | ✉ preetam.naik3@gmail.com

Test Date: August 17, 2025

Computer Science 94 /100	**Logical Ability** 72 /100	**Computer Programming** 74 /100	**Quantitative Ability (Advanced)** 80 /100
English Comprehension 64 /100	**WriteX - Essay Writing** 5 /100	**Automata** 75 /100	**Automata Fix** 43 /100
Personality ✓ Completed			

Computer Science

94 / 100

OS and Computer Architecture	DBMS	Computer Networks
86 / 100	94 / 100	100 / 100

Logical Ability

72 / 100

Inductive Reasoning	Deductive Reasoning	Abductive Reasoning
67 / 100	82 / 100	68 / 100

Computer Programming

74 / 100

Basic Programming	Data Structures	OOP and Complexity Theory
73 / 100	77 / 100	71 / 100

Quantitative Ability (Advanced)

80 / 100

Basic Mathematics	Advanced Mathematics	Applied Mathematics
85 / 100	76 / 100	80 / 100

English Comprehension

64 / 100

CEFR: **B2**

Grammar	Vocabulary	Comprehension
63 / 100	73 / 100	57 / 100

WriteX - Essay Writing

5 / 100

CEFR: **A1**

Content Score	Grammar Score
5 / 100	5 / 100

Automata

75 / 100

Programming Ability	Programming Practices	Functional Correctness
70 / 100	100 / 100	60 / 100

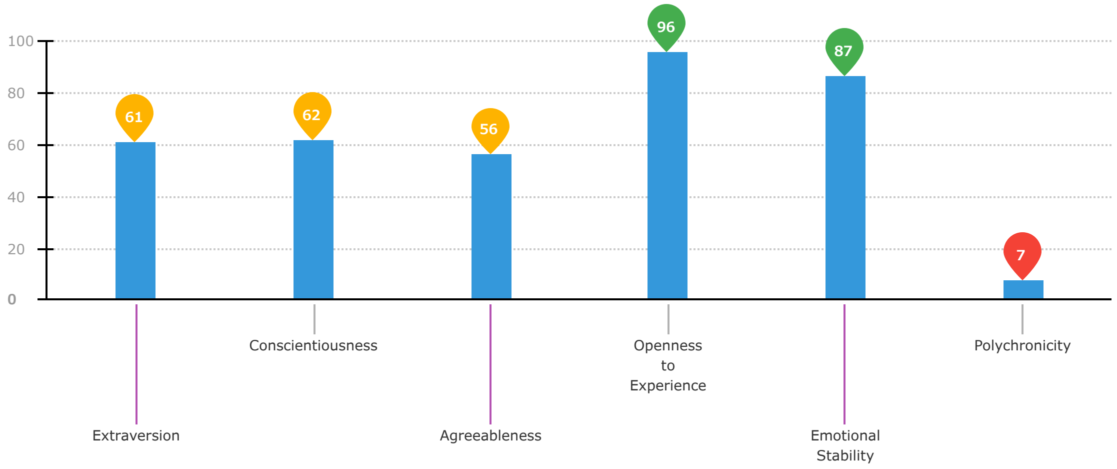
Automata Fix

43 / 100

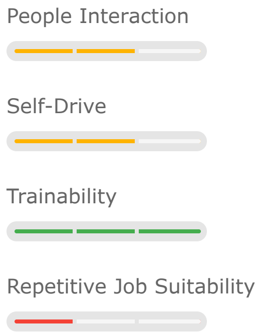
Code Reuse	Logical Error	Syntactical Error
50 / 100	25 / 100	100 / 100

Personality

Completed



Competencies



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

64 / 100 | CEFR: **B2**

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

You have a good understanding of commonly used grammatical constructs. You are able to read and understand articles, reports and letters/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

72 / 100

Inductive Reasoning

67 / 100

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

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

Deductive Reasoning

82 / 100

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

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

Abductive Reasoning

68 / 100

Quantitative Ability (Advanced)

80 / 100

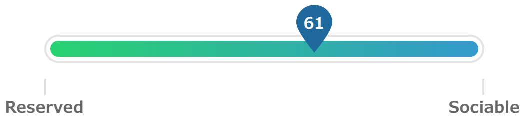
This test aims to measure your ability to solve problems on basic arithmetic operations, probability, permutations and combinations, and other advanced concepts.

It is commendable that you are able to understand and solve complex arithmetic problems. You are able to solve basic problems of probability, logarithms, permutations, and combinations. This skill will help you in jobs where one needs to work with statistical data and make probabilistic predictions.

Personality

Competencies

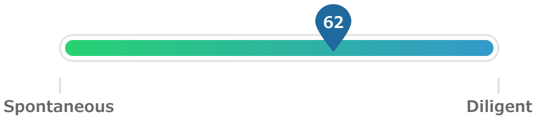
Extraversion



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

- You are 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.

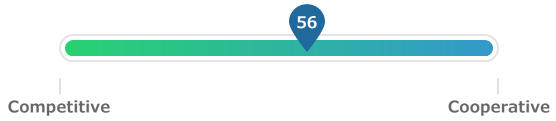
Conscientiousness



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.

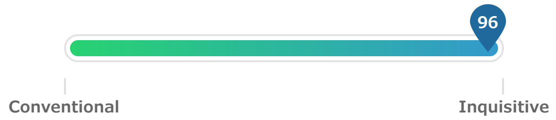
Agreeableness



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 flexible regarding your opinions and be willing to accommodate the needs of others.
- You are generally considerate of the needs of others yet may, at times, overlook social norms to achieve personal success.
- You are selective about the people you choose to trust.
- You are caring and you empathise a friend in distress.
- You give credit to others but also tends to be open with your friends about personal achievements.
- You are more inclined to strike a compromise in tough situations and may be suitable for jobs that demand managing expectations among different stakeholders.

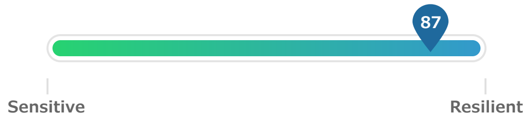
Openness to Experience



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.

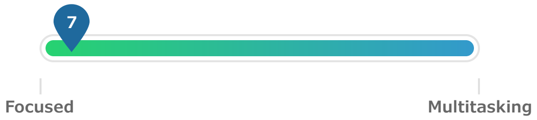
Emotional Stability



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

5 / 100 | CEFR: **A1**

Question

In your opinion, when is it justified for a company to fire its employees from the job? What are the considerations to be taken before making such a decision?
Support your response with reasons and suitable examples.

Scores

Content Score

5 / 100

Grammar Score

5 / 100

Response

in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high

Error Summary

●	Spelling	16
●	White Space	5
●	Style	0
●	Grammar	7
●	Typographical	1

Essay Statistics

236	1	236	142	108
Total words	Total sentences	Average sentence length	Total unique words	Total stop words

Error Details

Spelling

in my opinon it is justified for a company to fire i...	Possible spelling mistake found
...rmance or any violation of the rules is beeing done for exampke when a employee consis...	Possible spelling mistake found
...olation of the rules is beeing done for exampke when a employee consistently misses the...	Possible spelling mistake found
...e rules is beeing done for exampke when a employee consistently misses the deadli...	Use "an" instead of 'a' if the following word starts with a vowel sound, e.g. 'an article', 'an hour'
...sses the deadlines or does any sexually harasment or theft etc which creates a toxic work...	Possible spelling mistake found
... however the employee must be give fair oppurtunity to improve through warning during the p...	Possible spelling mistake found
...ce decrement or conduct any performance improment plans or workshops and give the employ...	Possible spelling mistake found
...tions follow a step by step termination proess which include everything mentioned abov...	Possible spelling mistake found
...oned above to ensure a fair termination proccess additionally companies recognize the hu...	Possible spelling mistake found
...e human impact of firing and offer any carrer suppot or pay whenever possible example...	Possible spelling mistake found
... impact of firing and offer any carrer suppot or pay whenever possible example while ...	Possible spelling mistake found
... other employee and reduce trust in the managment and ham the company reputation therefor...	Possible spelling mistake found
...efore any such decision made need to be handeled with transperency and respect towards t...	Possible spelling mistake found
... decision made need to be handeled with transperency and respect towards the employee by pro...	Possible spelling mistake found
...wards the employee by providing him job replacment assitance to soften the impact on the ...	Possible spelling mistake found
...mployee by providing him job replacment assitance to soften the impact on the employee a...	Possible spelling mistake found

White Space

...performance improment plans or workshops and give the employee clear expectations...	Possible typo: you repeated a whitespace
...ies recognize the human impact of firing and offer any carrer suppot or pay whene...	Possible typo: you repeated a whitespace
...on it should be taken care and evaluated with fairness and the needs should be pr...	Possible typo: you repeated a whitespace
...airness and the needs should be provided BUT considering all this companies shoul...	Possible typo: you repeated a whitespace
...eplacment assitance to soften the impact on the employee and keep the company rep...	Possible typo: you repeated a whitespace

Grammar

in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider replacing it with "employees".
in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider replacing it with "there".

in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider replacing it with "reasons".
in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider replacing it with "affect".

in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider replacing it with "organization's".
in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider replacing it with "done.".

in my opinon it is justified for a company to fire its employee only when they are clear reason that seriously effect the organization performance or any violation of the rules is beeing done for exampke when a employee consistently misses the deadlines or does any sexually harasment or theft etc which creates a toxic workplace to work in however the employee must be give fair oppurtunity to improve through warning during the performance decrement or conduct any performance improment plans or workshops and give the employee clear expectations to reach before the warnings or deadlines many organizations follow a step by step termination proess which include everything mentioned above to ensure a fair termination proccess additionally companies recognize the human impact of firing and offer any carrer suppot or pay whenever possible example while firing some may be in the need of medical support so to protect the health standard of the organization it should be taken care and evaluated with fairness and the needs should be provided BUT considering all this companies should know the impact of firing an employee it can lower the morale of other employee and reduce trust in the managment and ham the company reputation therefore any such decision made need to be handeled with transperency and respect towards the employee by providing him job replacment assitance to soften the impact on the employee and keep the company reputation high	Possible grammar error found. Consider removing "the" from here.

Typographical

in my opinon it is justified for a company...	This sentence does not start with an uppercase letter

Automata

● 75 / 100 Code Replay

Question 1 (Language: C)

High similarity detected : **97%**

An e-commerce company is planning to give a special discount on all its products to its customers for the holiday. The company possesses data on its stock of N product types. The data for each product type represents the count of customers who have ordered the given product. If the data K is positive then the product has been ordered by K customers and is in stock. If the data K is negative then the product has been ordered by K customers but is not in stock. The company will fulfill the order directly if the ordered product is in stock. If it is not in stock then the company will fulfill the order after they replenish the stock from the warehouse. They are planning to offer a discount amount A for each product. The discount value will be distributed to the customers who have purchased that selected product. The discount will be distributed only if the discount amount A can be divided by the number of orders for a particular product.

Write an algorithm for the sales team to find the number of products out of N for which the discount will be distributed.

Scores

Programming Ability

100 / 100

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

Programming Practices

100 / 100

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

Functional Correctness

100 / 100

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

Final Code Submitted

Compilation Status: Pass

```
//Header Files
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>

/* only used in string related operations */
typedef struct String string;
struct String
{
    char *str;
};

char *input(FILE *fp, int size, int has_space)
{
    int actual_size = 0;
    char *str = (char *)malloc(sizeof(char)*(size+actual_size));
    char ch;
    if(has_space == 1)
    {
        while(EOF != (ch=fgetc(fp)) && ch != '\n')
        {
            str[actual_size] = ch;
            actual_size++;
            if(actual_size >= size)
            {
                str = realloc(str,sizeof(char)*actual_size);
            }
        }
    }
    else
    {
```

Code Analysis

Average-case Time Complexity

Candidate code: O(N)

Best case code: O(N)

*N represents number of different types of products.

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

```
        while(EOF != (ch=fgetc(fp)) && ch != '\n' && ch != ' ')
        {
            str[actual_size] = ch;
            actual_size++;
            if(actual_size >= size)
            {
                str = realloc(str,sizeof(char)*actual_size);
            }
        }
    }
    actual_size++;
    str = realloc(str,sizeof(char)*actual_size);
    str[actual_size-1] = '\0';
    return str;
}
/* only used in string related operations */

typedef struct array_single_int array_single_int;
struct array_single_int
{
    int *data;
    int size;

};


/*
 *
 */
int  noOfProducts(array_single_int order, int disAmount)
{
    int  answer=0;
    for(int i=0;i<order.size;i++){
int customer=abs(order.data[i]);
if(customer>0 && disAmount%customer==0){
    answer++;
}
    }
    return answer;
}

int main()
{
    array_single_int order;
    int disAmount;

    //input for order
    scanf("%d", &order.size);
    order.data = (int *)malloc(sizeof(int) * order.size);
    for ( int idx = 0; idx < order.size; idx++ )
    {
      scanf("%d", &order.data[idx]);
    }

    //input for disAmount
    scanf("%d", &disAmount);


    int result = noOfProducts(order, disAmount);
    printf("%d", result);
    free (order.data);

    return 0;
}

```

Test Case Execution

Passed TC: **100%**

Total score

11/11

100%	**100%**	**100%**
Basic(**6**/6)	Advance(**2**/2)	Edge(**3**/3)

Compilation Statistics

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

Response time:	**00:08:43**
Average time taken between two compile attempts:	**00:02:11**
Average test case pass percentage per compile:	**25%**

Similarity Detected

Every response is checked against:

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

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

Candidate Response	Peer's Response
`{`	`{`
`  int *data;`	`  int *data;`

54	int size;	54	int size;
55			
56	};	55	};
57		56	
58		57	
...		...	
62	int noOfProducts(array_single_int order, int disAmount)	61	int noOfProducts(array_single_int order, int disAmount)
63	{	62	{
64	int answer=0;	63	int answer=0;
		64	// Write your code here
65	for(int i=0;i<order.size;i++){	65	for(int i=0;i<order.size;i++){
66	int customer=abs(order.data[i]);	66	int k=order.data[i];
67	if(customer>0 && disAmount%customer==0){	67	if(k>0 && (disAmount%k==0)){
68	answer++;	68	answer++;
69	}	69	}
70	}	70	}
		71	
		72	
71	return answer;	73	return answer;
72	}	74	}
73		75	
...		...	
90		92	
91	int result = noOfProducts(order, disAmount);	93	int result = noOfProducts(order, disAmount);
92	printf("%d", result);	94	printf("%d", result);
93	free (order.data);		
94		95	
95	return 0;	96	return 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++(G++9.3))

You are given a list of integers of size N. Write an algorithm to sort the first K elements (from list[0] to list[K-1]) of the list in ascending order and the remaining (list[K] to list[N-1]) elements in descending order.

Scores

Programming Ability

40 / 100

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

Programming Practices

100 / 100

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

Functional Correctness

20 / 100

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

Final Code Submitted

Compilation Status: Pass

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

Code Analysis

Average-case Time Complexity

```
#include<string>
#include<vector>
using namespace std;


/*
 *
 */
vector<int> funcSort (vector<int> inputList, int num)
{
   vector<int>answer=inputList;
  for(int i=0;i<num-1;i++){
     for(int j=i+1;j<num;j++){
      if(answer[i]>answer[j]){
         int temp=answer[i];
         answer[i]=answer[j];
         answer[j]=temp;

      }
     }
  }
   return answer;
}

int main()
{

   //input for inputList
   int inputList_size;
   cin >> inputList_size;
   vector<int> inputList;
   for ( int idx = 0; idx < inputList_size; idx++ )
   {
      int temp;
      cin >> temp;
      inputList.push_back(temp);
   }
   //input for num
   int num;
   cin >> num;


   //output
   vector<int> result = funcSort(inputList, num);
   for ( int idx = 0; idx < result.size() - 1; idx++ )
   {
      cout << result[idx] << " ";
   }
   cout << result[result.size() - 1];

```

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 logN)

*N represents number of elements in the input list

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

```
    return 0;
}

```

Test Case Execution

Passed TC: **28.57%**

Total score 2/7

0% Basic(**0**/5)	**100%** Advance(**1**/1)	**100%** Edge(**1**/1)

Compilation Statistics

16	15	1	0	0	5
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:25:30**
Average time taken between two compile attempts:	**00:01:36**
Average test case pass percentage per compile:	**13.39%**

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

43 / 100 | Code Replay

Question 1 (Language: C++)

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

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

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

.

Helper Description

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

```
class Point

{

   private:

   int X;

   int Y;

   double Point_calculateDistance(Point *point1, Point *point2)

   {

       /*Return the euclidean distance between two input points.

       This can be called as -

       *  If P1 and P2 are two points then -

       *  P1->Point_calculateDistance(P2);*/

    }

}
```

Scores

Final Code Submitted **Compilation Status: Fail**

```
// You can print the values to stdout for debugging
using namespace std;
int isTriangle(Point *P1, Point *P2, Point *P3)
{
    // write your code here
    int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1->y)+P3->y*(P1->y-P2->y);
    if(area==0){
```

Code Analysis

Average-case Time Complexity

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

Best case code:

```
        return 0;
    }
    else{
        return 1;
    }
}
```

*N represents

Errors/Warnings

```
In file included from main_25.cpp:8:
source_25.cpp: In function 'int isTriangle(Point*,
Point*, Point*)':
source_25.cpp:6:18: error: 'class Point' has no
member named 'x'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:25: error: 'class Point' has no
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:31: error: 'class Point' has no
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:38: error: 'int Point::X' is private
within this context
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
In file included from main_25.cpp:7:
ds_debugging_23.cpp:6:10: note: declared private
here
int X;
^
In file included from main_25.cpp:8:
source_25.cpp:6:38: note: field 'int Point::X' can be
accessed via 'int Point::Point_Return_X()'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
Point_Return_X()
source_25.cpp:6:45: error: 'class Point' has no
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:51: error: 'class Point' has no
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:58: error: 'class Point' has no
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:65: error: 'class Point' has no
```

```
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
source_25.cpp:6:71: error: 'class Point' has no
member named 'y'
int area=P1->x*(P2->y-P3->y)+P2->X*(P3->y-P1-
>y)+P3->y*(P1->y-P2->y);
^
```

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:	**00:05:20**
Average time taken between two compile attempts:	**00:05:20**
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++)

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

```
// You can print the values to stdout for debugging
using namespace std;
float allExponent(int baseValue, int exponentValue)
{
    float res = 1;
    if(exponentValue >=0)
    {
        res = (float)positiveExponent(baseValue, exponentValue);
    }
    else
    {
        // write your code here for negative exponentInput
        res=1.0/(float)positiveExponent(baseValue,-exponentValue);
    }
    return res;
}


```

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	Basic	Advance	Edge
6/6	**100%** Basic(**2**/2)	**100%** Advance(**3**/3)	**100%** Edge(**1**/1)

Compilation Statistics

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

Response time:	**00:01:34**
Average time taken between two compile attempts:	**00:01:34**
Average test case pass percentage per compile:	**100%**

Average-case Time Complexity

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

Test Case Execution

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

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

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

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

Question 3 (Language: C++)

The function/method ***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

```
// You can print the values to stdout for debugging
void selectionSortArray(int len, int* arr){
  for(int x=0; x<len-1; x++){
    int index_of_min = x;
    for(int y=x+1; y<len; y++){
      if(arr[y]<arr[index_of_min]){
        index_of_min = y;
      }
    }
    int temp = arr[x];
    arr[x] = arr[index_of_min];
    arr[index_of_min] = 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

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

6	6	0	5	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:02:46**
Average time taken between two compile attempts:	**00:00:28**
Average test case pass percentage per compile:	**25.9%**

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 4 (Language: C++)

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

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

Scores

Final Code Submitted **Compilation Status: Pass**

```
// You can print the values to stdout for debugging
using namespace std;
int matrixSum(int rows, int columns, int **matrix)
{
  int sum=0;
  for(int i=0;i<rows;i++)
  {
    for(int j=0;j<columns;j++)
      sum += matrix[i][j];
  }
  return sum;
}
```

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	100%	100%	0%
10/10	Basic(**6**/6)	Advance(**4**/4)	Edge(**0**/0)

Compilation Statistics

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

Response time:	**00:02:02**
Average time taken between two compile attempts:	**00:02:02**
Average test case pass percentage per compile:	**100%**

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

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

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

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

Scores

Final Code Submitted

Compilation Status: Pass

```
// You can print the values to stdout for debugging
#include <iostream>
using namespace std;
void patternPrint(int num)
{
    int print=1;
    for(int i=0;i<num;i++)
    {
        for(int j=0;j<=i;j++);
        {
            cout<<print;
            print++;
        }
        cout<<"\n";
    }
}

```

Code Analysis

Average-case Time Complexity

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

Best case code:

*N represents

Errors/Warnings

There are no errors in the candidate's code.

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Test Case Execution

Passed TC: **25%**

Total score 2/8

14%	0%	100%
Basic(**1**/7)	Advance(**0**/0)	Edge(**1**/1)

Compilation Statistics

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

Response time:	**00:02:53**
Average time taken between two compile attempts:	**00:01:27**
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 6 (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: Fail

```
// You can print the values to stdout for debugging
```

Code Analysis

```
 2 #include<iostream>
 3 using namespace std;
 4 void descendingSortArray( int* arr)
 5 {
 6   int temp=0;
 7   int len=arr.size;
 8   for(int i=0; i<=len-1;i++)
 9   {
10     for(int j=len-1; j<=0;j++)
11     {
12 
13       if(arr[i]>arr[j])
14       {
15         temp=arr[i];
16         arr[i]=arr[j];
17         arr[j]=temp;
18       }
19     }
20   }
21 }
```

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_10.cpp:5:
source_10.cpp: In function 'void
descendingSortArray(int*)':
source_10.cpp:7:17: error: request for member 'size'
in 'arr', which is of non-class type 'int*'
int len=arr.size;
^~~~
main_10.cpp: In function 'int main(int, const char**)':
main_10.cpp:18:29: error: invalid conversion from
'int' to 'int*' [-fpermissive]
descendingSortArray(n, arr);
^
main_10.cpp:18:35: error: too many arguments to
function 'void descendingSortArray(int*)'
descendingSortArray(n, arr);
^
In file included from main_10.cpp:5:
source_10.cpp:4:6: note: declared here
void descendingSortArray( int* arr)
^~~~~~~~~~~~~~~~~~~
```

Structural Vulnerabilites and Errors

There are no errors in the candidate's code.

Compilation Statistics

6	1	5	1	0	0
Total attempts	Successful	Compilation errors	Sample failed	Timed out	Runtime errors

Response time:	**00:03:51**
Average time taken between two compile attempts:	**00:00:39**
Average test case pass percentage per compile:	**10%**

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++)

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

```
// You can print the values to stdout for debugging
#include<string.h>
using namespace std;
int checkBirthDay(char* month, int day)
{
    if((strcmp(month,"July")) || (day==5))
            return 1;
        else
            return 0;
}

```

Code Analysis

Average-case Time Complexity

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

Best case code:

*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: **33.33%**

Total score

3/9

0%	**100%**	**0%**
Basic(**0**/6)	Advance(**3**/3)	Edge(**0**/0)

Compilation Statistics

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

Response time:	**00:00:14**
Average time taken between two compile attempts:	**00:00:00**
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

4 | Learning Resources

English Comprehension

Improve your vocabulary of terms used in business	$	🌐	📖
Improve your knowledge of Business English	FREE	🌐	📖
Improve your hold on the language by reading Shakespearan plays	FREE	🌐	📖

Logical Ability

Practice your Inductive Reasoning Skills!	$	🌐	📖
Take a course on advanced logic	FREE	🌐	🎬
Test your deduction skills!	FREE	🌐	📖

Quantitative Ability (Advanced)

Learn about the application of Bayes' Theorem in varied fields	FREE	W	📖
Learn about Fermet's last theorem	FREE	W	📖
Learn about the principles of statistics	$	🌐	📖

Icon Index

FREE	Free Tutorial	$	Paid Tutorial	▶	Youtube Video	🌐	Web Source
▶	Wikipedia	📖	Text Tutorial	🎬	Video Tutorial	▷	Google Playstore