### B Tech Computer Science and Engineering (Data Science)



### -311 Program Semester – II

**Academic Year 2023-24**

**Object Oriented Programming and Design**

### Mini Project

#### Aptitude Test Analyzer

Developed by:

L004 Ishaa Bhalekar L005 Arpita Das L007 Sini Goyal L014 Saanvi Kanodia

**Faculty In charge Dr. Dhirendra Mishra Dr. Priteegandha Naik**

### SVKM’s NMIMS

### Mukesh Patel School of Technology Management and Engineering Vile Parle West Mumbai-56

1

Contents



[Abstract 3](#_bookmark0)

[Introduction 3](#_bookmark1)

[Scope and societal need 4](#_bookmark2)

[Organisation Of Project 5](#_bookmark3)

[Research Question 6](#_bookmark4)

[Block diagram 7](#_bookmark5)

[Problem Statement 8](#_bookmark6)

[List of classes and its members 9](#_bookmark7)

[Use case diagram 11](#_bookmark8)

[Class diagram 13](#_bookmark9)

[Activity diagram 15](#_bookmark10)

[Sequence diagram 17](#_bookmark11)

[Object oriented concepts 19](#_bookmark13)

[Algorithm 21](#_bookmark15)

[Flowchart diagram 23](#_bookmark17)

[Code in C++: 25](#_bookmark18)

[Output screenshot 3](#_bookmark21)3

[List of errors obtained and its solutions 4](#_bookmark22)1

[Scope of improvement 4](#_bookmark23)2

[References 42](#_bookmark24)

2

**1.**



# Abstract**:**

Our ‘Aptitude Test Analyser’ program uses C++ coding language to create a testing and analysing application that will allows users to find their strengths, weaknesses, and overall prowess in different domains i.e. math, logic, critical thinking, and general knowledge.

We have tested out multiple such aptitude tests and noticed that the biggest drawback to all of them was that they charged a price or required subscriptions to be able to access the analysis report of the test. With this in mind, we wanted to create a software that could provide necessary analysis of the user’s performance by gauging skills like aptitude, accuracy, and speed. By using concepts of Object-Oriented Programming, we aim to imitate the format of online tests while adding additional features and creating a user- friendly environment and free-to-use program for users. To do so we utilize classes and source files, which segregates various types of data into separate sections to prevent unintended modifications and manipulations. This helps limit data manipulation to specific functions that are granted appropriate access privileges. This approach ensures a structured and secure environment for managing and accessing data within the program.

**2.**

# Introduction**:**

An aptitude test app is one that provides users/testers with a set of questions based on different domains. The user’s performance in the test will help determine their strength and weakness area for every domain. Such tests help users find out what their currents skills are and the areas that they lack in and need to work on. These questions are well rounded and test a person’s verbal ability, numerical ability, spatial ability, mechanical ability, logical reasoning, clerical speed, and accuracy. These tests are used by different kinds of users for different purposes. Students take such tests either to know their career path, or when they appear for aptitude tests as a requirement for admission in colleges. Similarly, job-seekers appear for such tests and are judged and hired based on their performance. Different organizations will give different questions based on what skills they require in their organization. The currents apps and software that exist are extremely advanced with their high-quality user experience, convenience and with multiple features like trackers, graphical report of the analysis, animations, and personalization features.

While there are many projects that create test analysis apps using various framework/stylistic languages, there are hardly any publicly available apps that provided such analysis for free. Therefore, our ‘aptitude test analyzer’ program aims to provide users with a platform to test their aptitude skills and get a detailed analysis of four different sections like Critical thinking, Problem Solving, Logical reasoning and Mathematical application.

3

**3.**



# Scope and societal need

Recently, people have expressed a higher interest in creating career and educational decisions on where scientific grounds. Many people put their trust into aptitude tests to tell them accurately what skill a strength for them is and which skills they can turn into a profession. Students, and jobseekers alike, can evaluate potential career choices based on their strengths and weaknesses by taking an aptitude test. The analysis they receive helps them identify weaker areas and improve them. The program guarantees an in-depth and open examination of the test a sound background. Since the program we created is free, more students are encouraged to make informed decisions by getting to know their aptitude analysis. The test ensures that all students have equal and accessible opportunities to implement and evaluate their skills

4

**4.**



# Organisation Of Project

* 1. Storyline -

To design an interactive program that allows users to take a test in multiple sections and provides a detailed analysis, feedback and performance metrics which includes speed of answering, accuracy of correct answers, section wise time taken and strength/weakness pertaining to the sections. The test consists of 4 sections: Problem Solving, Critical Thinking, Logical and Mathematical. Each section of the test has total 5 questions. The answers will be given by the user to all the 20 questions. And then time taken for completing the entire test will be shown. After which two three choices will be shown: Review / Analyse / Exit.

* 1. **Variables**

**Inputs-**

* + 1. User can input answer.
    2. User can input their choice of options from Analysis, review, and exit

Outputs –

1. Instructions for the test
2. Section-wise time
3. Total time taken.
4. Section wise time
5. Marks obtained out of 20
6. Review of the entire test
7. Strength and weakness analysis of every domain

**Constraints –** Section wise Timer

1. Until user does not correctly input the key word ‘ready’ the test will not begin, and if wrong word is input, then the instruction page will keep looping.0
2. Time limit for answering each section of the test is set to 30 seconds. Once this time allotted is over, the test will automatically end.
3. User's answer choices are limited to options 'a', 'b', 'c', or 'd' for each question.

5

**5.**



# Research Question

* 1. Research Question**:**

"How effective is an automated aptitude test analyser in evaluating users' aptitude across different domains?"

This research question focuses on assessing the effectiveness of the developed software in analysing users' aptitude based on their performance in arithmetic, logical, series completion, and miscellaneous domains. It involves evaluating the accuracy of the analyser in calculating scores, analysing the speed and accuracy of user responses, and identifying strengths and weaknesses in each domain. Additionally, it may explore user satisfaction and feedback regarding the usability and usefulness of the software.

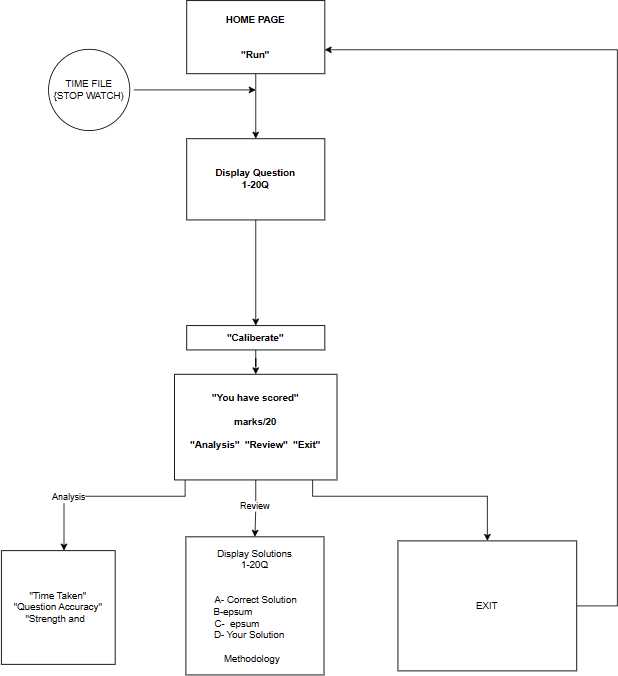
* 1. Research Objective:

The aim of the project is to create an aptitude test for students that provides a detailed performance analysis based on tested skills respectively. Additionally, we aim to integrate these features with a constant running timer block that will help the user to understand the time taken to complete the test. The aptitude test will assess the tester’s presence of mind and expertise in a particular domain. It will help them evaluate their progress and control over time by comparing their test scores and speeds in different domains like critical thinking, logical reasoning, and problem- solving skills respectively.

6

# **6.**

# Block diagram



* 1. Preface: This diagram gives the broad diagrammatic overview of all the main functions that will be carried out and how they will be displayed in the console on user side.

Fig 6.1: block diagram

7

# **7.**

# Problem Statement



Develop an ‘Aptitude Test Analyzer’ program in C++ which will print 20 aptitude-based MCQ questions in total. There should be 5 questions each in the 4 domains/sections: ‘Arithmetic’, ’Logical’, ’Series Completion’ and ‘Miscellaneous’. Take the user’s answer to every answer and move to the next question, only if appropriate value is printed. After every section finishes, print the time remaining in the test, and the time taken to solve that section. There should be a condition that if the user is not able to complete answering the section within the given time limit, then the test will end on its own.

Upon completion, the program provides users with a detailed analysis of their performance, including total time taken, accuracy, section-wise marks, and identification of strengths and weaknesses.

The following functionalities should be included:

* + 1. Display instructions about the test format, time limits.
    2. Utilize a timer mechanism to track the time remaining for each section, with intervals for updating the user interface.
    3. Record user answers and calculate their total score based.
    4. Provide users with the option to review their answers, view detailed analysis reports, and identify areas for improvement.
    5. Display section-wise time taken by the user and evaluate their speed in answering questions within each domain.

8



**8.**

# List of classes and its members

1. database class:
   * Protected Members:
     + string method[20]: stores methods to solve each question.
     + string ans[20]: stores correct answers of each question.
     + string quest[100] : stores questions.
   * Public Members:
     + string qdup[100] : stores a copy of the questions.
     + string instructions() : Method to display test instructions and take user's readiness.
     + void print\_quest() : Method for printing questions from the database file.
     + void print\_method(int n) : Method to print the method of every question.
2. User class (inherits from database):
   * Protected Members:
     + string notes[20] : Array to store user's notes for each question.
   * Public Members:
     + void store\_ans(int i) : Method to store user's answer for a question.
3. Timer class(inherits from User ):
   * Public Members:
     + clock\_t start[4] : Array to store start time for each section.
     + clock\_t stop[4] : Array to store stop time for each section.
     + float total\_time : Total time taken by the user to complete the test.
   * Public Methods:

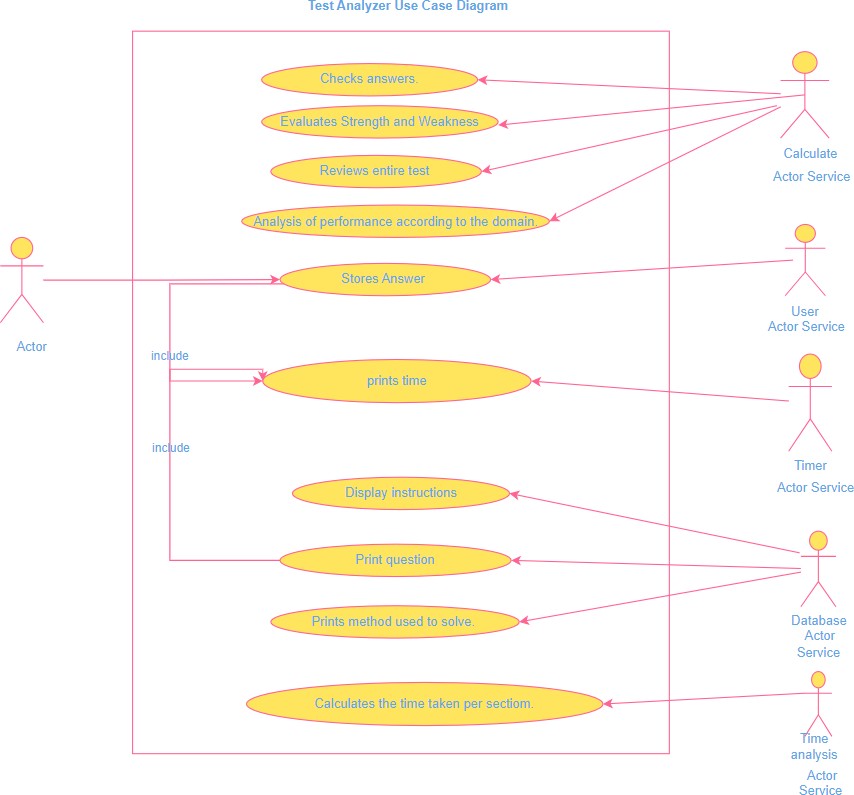
9

* + - void print\_time() : Method to print time intervals, calculate section-wise time, and total time taken.



1. Calculate class (inherits from User ):
   * Protected Members:
     + int score[20] : Array to store user's scores for each question.
     + int sect\_mrks[4] : Array to store section-wise marks.
     + int total : Total score of the user.
     + double accuracy : Accuracy percentage of the user.
     + string indicator : Indicator of strength or weakness for each section.
   * Public Members:
     + void review() : Method to review user's answers, correct answers, and explanations.
     + int check() : Method to compare user's answers with correct answers and calculate the total score.
     + void analysis() : Method to analyze user's performance, calculate accuracy, section- wise marks, and identify strengths and weaknesses.
     + void strength\_weak() : Method to identify strengths and weaknesses based on section-wise marks.
2. Time\_Analysis class(inherits from Timer ):
   * Public Members:
     + void section\_time() : Method to print section-wise time taken by the user.

10



**9.**

# Use case diagram

* 1. Preface: the diagram displays the relationship between the actors and member function, it should which class will be performing which functionality and if the functionalities interact or include each other.

Fig 9.1: use case diagram

11



* 1. Description for Use Case Diagram:

1. Actors:

* The primary actor is the User (presumably a test-taker or an instructor).
* Other actors include:
* Calculate Actor Service: Responsible for storing answers.
* Time Actor Service: Manages timing-related tasks, such as starting the timer and calculating remaining time left for solving the remaining questions of that section.
* Database Actor Service: Handles displaying questions and printing methods used to solve them.
* Timer Actor Service: Calculates the time taken by the user.

1. **Use Cases** (Actions Performed by the User):

* Checks Answers: The user can verify their answers against the correct ones.
* Evaluates Strength and Weaknesses: Assess their performance in different areas.
* Reviews Entire Test: Allows a comprehensive review of the entire test.
* Analysis of Performance According to the Domain: Provides insights based on the test domain.

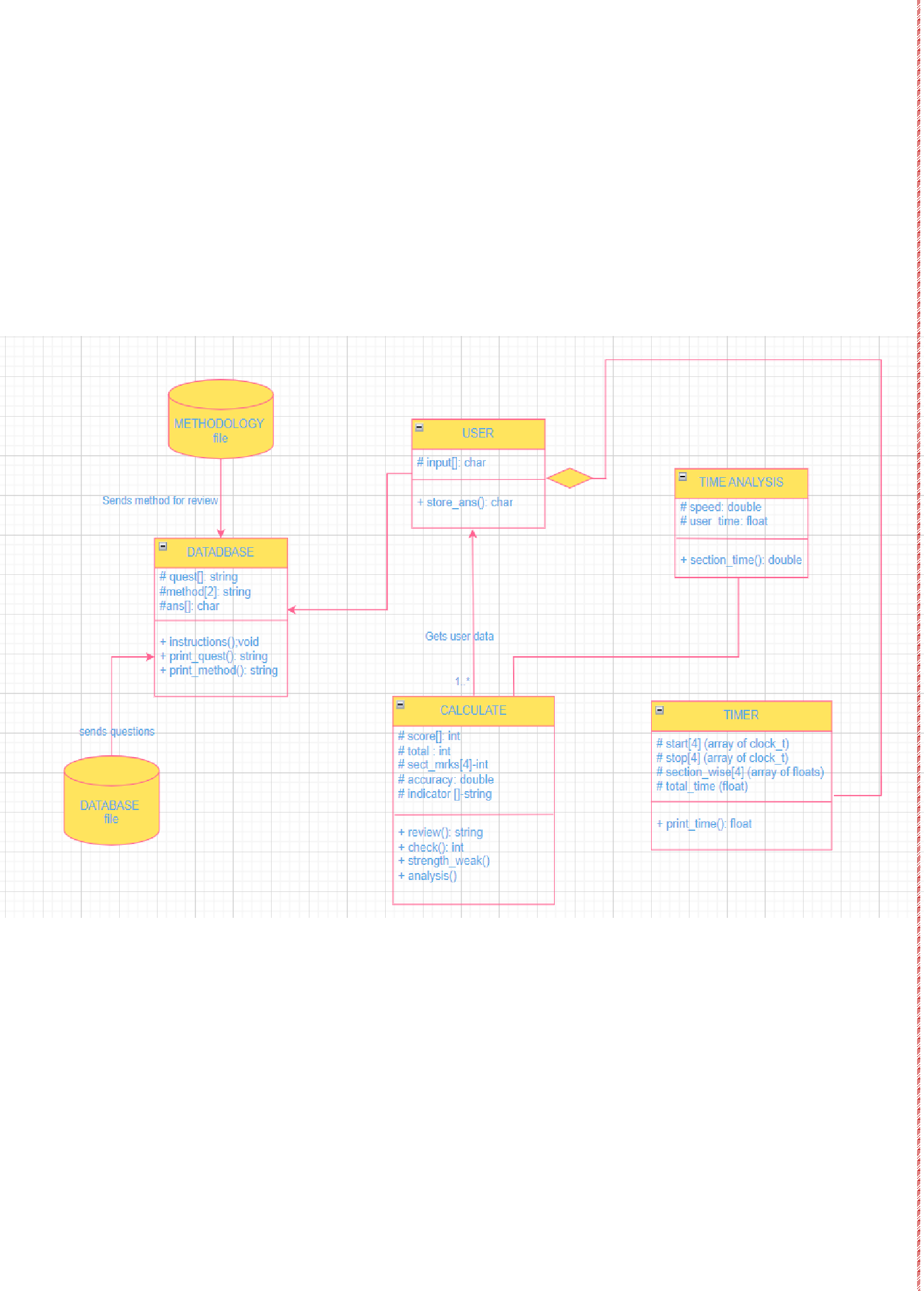
1. Interactions:

* The user interacts with the system through these use cases.
* The services (actors) assist in various tasks, such as giving answers, managing time, and displaying relevant information.

This diagram captures the high-level interactions between the user and the system, emphasizing functionality from the user’s perspective.

12

**10.**



# Class diagram:

**10.1** preface: Class diagram shows the members of every class and how the classes are related to each other. It shows all the components and files involved in every d class. It also shows the cardinality of every class with respect to the other. The arrows describe the direction of data transfer between classes and what is being primarily shared.

Fig 10.1: class diagram

* 1. Description of Class Diagram:

a. DATABASE Class:

* Attributes: question (string), method (string), ans1 (char).
* Methods: instruction() (void), print\_quest() (returns a string), print\_method() (returns a string).
* Purpose: The DATABASE class seems to store information related to questions, their solution methods, and answers.

13

b. Calculate Class:



* Attributes: score (int),total (int) sect\_mrks[4] (int array), accuracy (double), indicator[] (string array).
* Methods: review() (returns a string), check() (returns an int), strength\_weak(), analysis().
* Purpose: The Calculate class likely performs calculations based on data from the DATABASE and the user. It checks user input and provides analysis.

c. USER Class:

* Attributes: input1 (char).
* Methods: store\_ans() (returns a char).
* Purpose: The USER class interacts with the Calculate class, providing input and receiving feedback.

d. TIMER CLASS:

* Attributes: time (float), total\_time (float).
* Method: print\_time() (returns a float).
* Purpose: The TIMER class displays time information to the user and provides timing data for the TIME ANALYSIS.

e. TIME ANALYSIS CLASS:

* Attributes: speed (double), user\_time (float).
* Methods: , section\_time() (both return double values).
* Purpose: The TIME ANALYSIS class analyzes timing data received from the TIMER related to the user’s performance or actions.

14

**11.**



# Activity diagram

* 1. Preface: Activity diagram shows the flow of functionalities in sequential order.

A diagram of a process

Description automatically generated

11.1: activity diagram

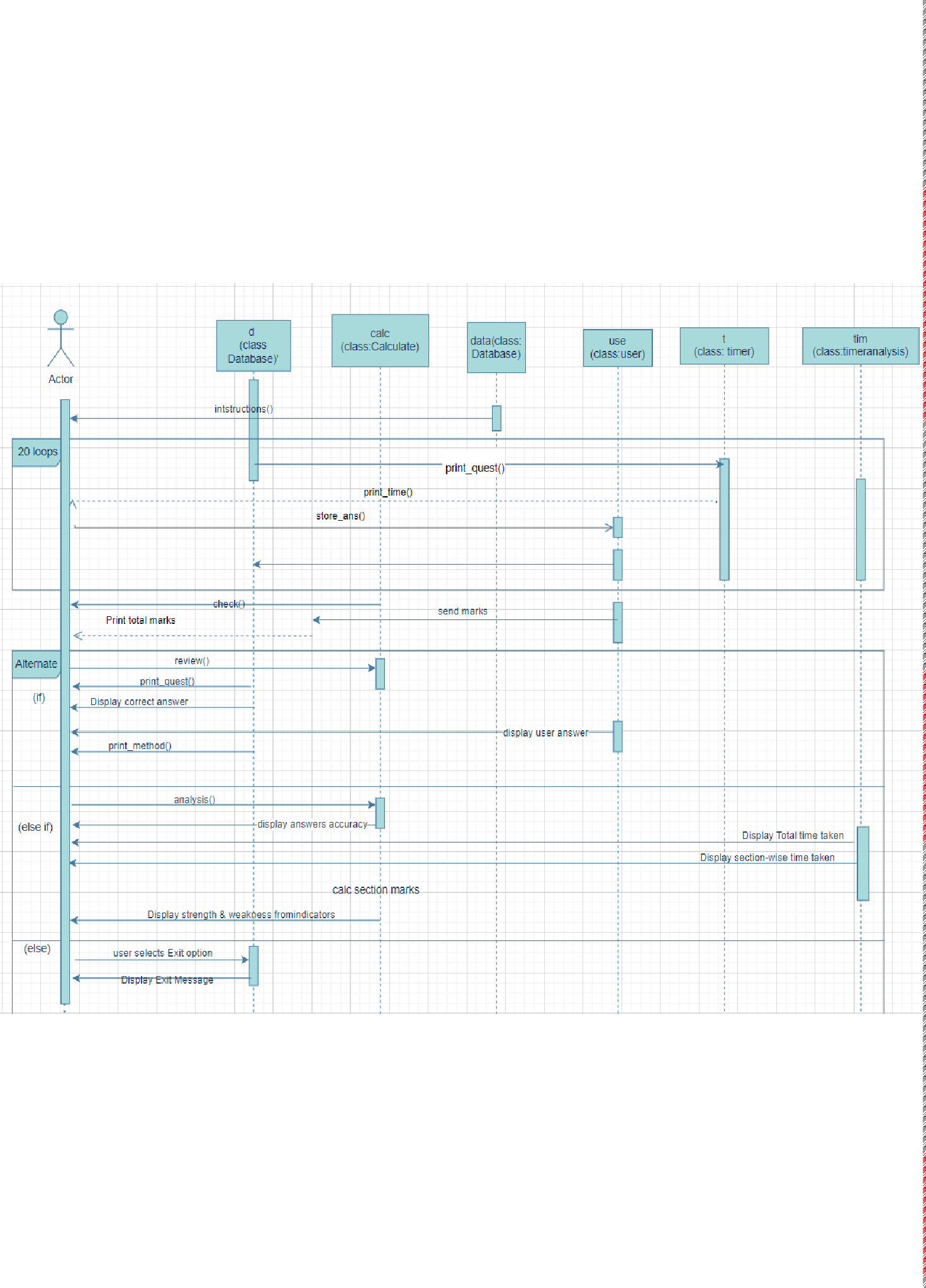
15



* 1. Description for Activity Diagram:
     1. Displaying Questions and Timer:
     2. When user will start the Test, the system displays a question for you to answer.
     3. Simultaneously, a timer begins counting down. It will be displayed at the top of the screen.
     4. Recording Your Answering Time:
     5. While answering the question the system records the time. This takes place for each question which is 20 times.
     6. Storing Your Answer:
     7. After answering each question, the user’s response is stored by the system.
     8. Viewing Total Marks and Accuracy:
     9. Next, the system displays the user's total marks and accuracy based on the user's answers.
     10. Choosing Between Review and Analysis:
     11. Users can decide whether to review their performance or analyze it further.
     12. Review Option:
     13. If the user chooses to review, the system shows you the questions along with their correct answers.
     14. Users can also view any notes they have entered and even print them if needed.
     15. Analysis Option:
     16. Alternatively, if user selects analysis, the system provides additional insights:
     17. Section-wise time distribution
     18. Strengths and weaknesses based on the user’s performance.
     19. Exiting the Activity:
     20. Finally, when the user is done reviewing or analyzing, you can exit the activity.

16

**12.**



# Sequence diagram

* 1. Preface: The sequences diagram shows the communication between the objects of the classes, and which functions they pass or call in a sequential order

Fig: 12.1: Sequence diagram

* 1. Description for Sequence Diagram:

When the program starts, the screen displays the homepage and instructions for the test. The user selects the "ready" option to begin. The first question is displayed, along with a timer indicating

17

the total time remaining in the test. This process repeats for 20 questions. Accepts the answer for 5 questions of each section and prints the time left for answering the remaining question. Upon completion, the screen displays the total marks and accuracy of the test. Accuracy is determined by the number of correct answers out of the total questions.



[**4**](#_bookmark28)

After completing user is presented with three options:

the test, the

* View Review: Displays each question, the correct answer, the user's answer, the methodology, and any previously created notes.
* View Analysis: Shows the total time taken to complete the test, shows the marks scored in each section and based on which section is a strength of the user, and which is a weakness.
* Exit: Allows the user to exit the program. After selecting this option, an exit page is displayed.

18

**13.**



# Object oriented concepts

* 1. Preface for concept:

Object Oriented Programming Language helps the programmer inculcate real-world phenomena while creating a program. The main purpose of OOPS is to combine a program's attributes and functionalities together.

* 1. Concepts
     1. Data Encapsulation: Through class the user can bind the data and its associated functions together. It allows the data (and functions) to be hidden, if necessary, from external use. When defining a class, we are creating a new abstract data type that can be treated like any other built in data type [1]. The project uses class, because it provides the user with features like readability, accessibility, repetition of variable names in multiple classes and allows users to have unambiguous code. [1, p. 99]
     2. Data Abstraction: Concept of hiding the complex implementation details and showing only the essential features of the object. Abstraction is applied in the code by exposing only necessary methods to interact with objects, such as store\_ans() to store user's answers or analysis() to analyze user's performance, while hiding the internal workings of these methods. [1]
     3. Inheritance: Inheritance is the mechanism of deriving a new class from an old one and is called inheritance. The old class is referred to as a base class and the new one is called derived/subclass. [1, p. 213] In the project different types of inheritance, specifically multilevel inheritance in class Calculate where we have inherited a class user and virtual base class named database. Also, class Timer where we derived the class Calculate and class Timer in class Time Analysis
     4. File stream: File streams allow handling of input and output operations with files. They allow us to read data from files using ifstream and write data to files using ofstream or, perform both operations using fstream [2, p. 325]. In our program, we are using a file called database.txt to print questions and methodologies. Objects are used to read and write into the file. [1, p. 338]

19

**13.2 Features**

* + - 1. File input/output (i/o):

The program reads questions and methods from external files (database.txt and method.txt) using file input operations (ifstream) and displays them to the user.

* + - 1. Multi-threading:

The program utilizes multi-threading to implement a timer mechanism for each section of the

, ensuring that the timer updates at regular intervals while the user interacts with the test.

* + - 1. Input validation:

Input validation is implemented to ensure that users provide valid answers (options 'a', 'b', 'c',

[**3**](#_bookmark27)

the user to enter

the program

invalid input is

or 'd') for each question. If an option.

a valid

provided,

prompts



* + - 1. Time-based analysis:

The program includes features to analyze the time taken by the user to complete each section of the test, providing insights into the user's speed of answering questions within each domain.

* + - 1. User interaction:

The program interacts with the user by displaying instructions, presenting questions, recording user responses, providing analysis reports, and allowing users to review their answers and exit the test.

* + - 1. Data storage and manipulation:

The program stores questions, methods, user answers, and other relevant data in arrays or variables, manipulates them during the test, and performs calculations to analyze the user's performance.

* + - 1. Modular programming:

The code is organized into classes representing different functionalities (e.g., database management, user interaction, timer implementation, analysis, etc.), promoting modular programming principles for better code organization, reusability, and maintenance.

13.4 Conclusion:

This program has used the object-oriented programming language's concept to implement all the necessary functionalities of the test. This has ensured the smooth functioning of the program by keeping members secured within their class, reducing manipulation. This allows the program to provide the desired output.

20



**14.**

# Algorithm

* 1. Start
  2. Display the instructions at the beginning of the test.
  3. If user enters ready go to step 4, else goto step
  4. Read int i =1 and begin with the first question.
  5. If there are more questions (i < 20), go to step 6; otherwise, if all questions are answered (i = 20), move to step 14.
  6. Display the current test question.
  7. Start the timer simultaneously as the user enters ready ,to keep track of section-wise time .
  8. Answer the question and record the answer.
  9. Display remaining time for answering unanswered questions.
  10. Displays time taken to solve each section.
  11. Repeat steps 6-10 until all questions are answered.
  12. Finish the test
  13. Display total time taken to solve the entire test.
  14. Calculate and display the total marks obtained.
  15. Provide choice to user to choose from review answers, analysis, or exit the test.
  16. If user select the review option, proceed to step 17; if analysis, go to step 19; if exit, move to step .
  17. User selects the review section:
      1. Display each question along with your answer, correct answer, method explanation, and any notes created by the user.
      2. If there are more questions to review, return to step 17.a.; otherwise, continue to step 18.
  18. End of review.
  19. Else user selects analysis section:
      1. Display the total time taken for the test.
      2. Calculate the accuracy of correct users

21

* Divide the test into sections (e.g., English, Math, Logic, etc.).
* Calculate the time taken for each section by subtracting the start time of the section from the end time.
* Display the time taken for each section.

1. Analyze your performance.
   * Count the number of correct answers in each section.

[**1**](#_bookmark25)

* + Determine the accuracy of each section

of correct answers

by dividing the

in that section.

by the total questions

number

* + Identify strengths and weaknesses based on the accuracy of each section.
  + Display the strengths and weaknesses.
  1. User selects analysis section:
     1. Display the total time taken for the test.
     2. Calculate the accuracy of correct users
* Divide the test into sections (e.g., English, Math, Logic, etc.).
* Calculate the time taken for each section by subtracting the start time of the section from the end time.
* Display the time taken for each section.

1. Analyze your performance.
   * Count the number of correct answers in each section.

[**1**](#_bookmark25)

of correct answers

by dividing the

* + Determine the accuracy of each section

in that section.

by the total questions

number

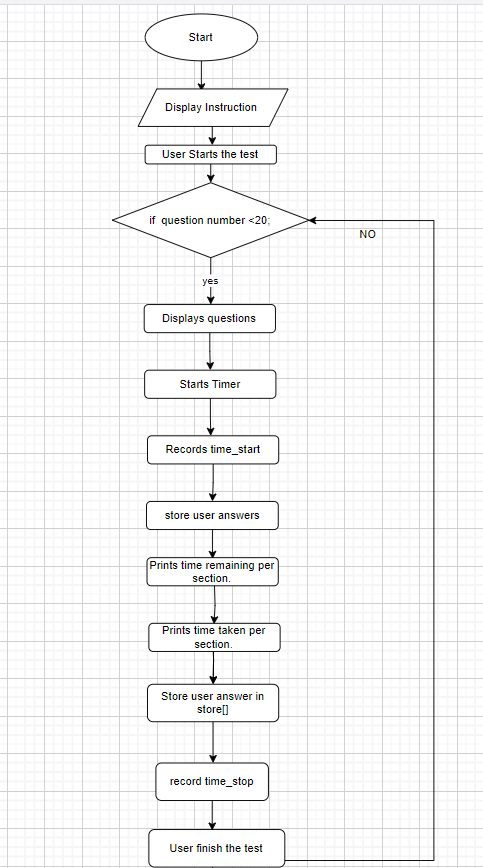
* + Identify strengths and weaknesses based on the accuracy of each section.



* + Display the strengths and weaknesses.
  1. End of analysis
  2. Exit
  3. Stop.

22

**15.**



# Flowchart diagram

**15.1 Preface:** The flowchart diagram represents the algorithm diagrammatically. It shows the detailed pseudocode flow of the entire program.

23

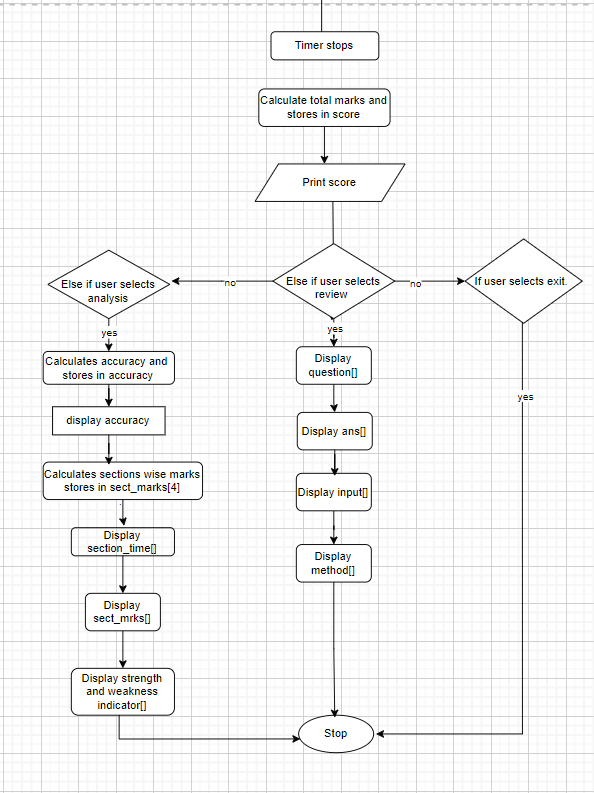


Fig 15.1: Flowchart

24



**16.**

# Code in C++:

#include <ctime>

#include <fstream>

#include <iostream>

#include <string>

#include <cstdlib>

#include <thread>

#include <chrono>

using namespace std;

string domain[4]={"Arithmetic","Logical","Series Completion", "Problem Solving"};

class database

{

protected:

string method[20];

string ans[20] = {"c", "b", "c", "c", "a", "d", "b", "c", "b", "c",

"c", "d", "a", "c", "a","a", "d", "a", "c", "c"

};

string quest[100];

public:

string qdup[100];

string instructions()

{

string reply;

cout << "\n\n\n\n\t\t\t\t-------------- WELCOME TO TEST ANALYZER APTITUDE TEST --------------- " << endl

<< endl;

cout << "\t\t\t\t---This test will analyze your aptitude in 4 domains: ---\n"<< endl;

cout << "\t\t\t\tInstructions:"<< endl;

cout << "\t\t\t\tThere are 20 MCQ questions in total "<< endl;

cout << "\t\t\t\tThe duration of the test is 5 minutes"<< endl;

cout << "\t\t\t\tA timer will show you the amount of time remaining after every question solved"<< endl;

cout << "\t\t\t\tOnce the time elapses the test will end automatically"<< endl;

cout << "\t\t\t\tOnce you have gone to the next question, you can't go back."<< endl;

cout << "\t\t\t\tTo progress to the next question, answer the question by typing the appropriate option letter"<< endl;;

cout << "\t\t\t\tType 'ready' once you are ready to begin the test" << endl;

cin>>reply;

return reply;

cout<<"\n\n\n";

}

25

float n1 = 1;



int n = 1;

void print\_quest()

{

ifstream in;

in.open("database.txt");

if (in.fail())

{

cout << "Couldn't open file";

exit(0);

}

else

{

for (int i = 0; i < 100 && !in.eof(); ++i)

{

getline(in, quest[i]);

qdup[i] = quest[i];

}

do

{

cout << qdup[n-1];

cout << endl;

n1 = n % 5;

n++;

}

while (n1 != 0);

}

in.close();

}

void print\_method(int n)

{

int i=0;

ifstream in;

in.open("method.txt");

if (in.fail())

{

cout << "Couldn't open file";

exit(0);

}

else

for (i; i < 20 && !in.eof(); i++)

{

string method2[20];

getline(in, method2[i]);

method[i]=method2[i];

// Close the file after reading

}

cout<<method[n];

cout<<endl;

26

in.close();



}

} d;

string input[20];

class User : public database

{

public:

void store\_ans(int i) // store user answer

{

bool val = true;

while(val)

{

cin.ignore();

cout << "Enter the answer : ";

cin >> input[i];

if(input[i]=="a"||input[i]=="b"||input[i]=="c"||input[i]=="d")

{

val=false;

}

else

cout<<"Enter valid option only, numerical, or any other answer will not be accepted"<<endl;

}

}

};

float section\_wise[4];

class Timer : public User

{

public:

clock\_t start[4];

clock\_t stop[4];

float total\_time;

float speed;

27



public:

void print\_time() // will print at intervals using threads

{

const int TIME\_LIMIT = 300;

total\_time = 0;

int c=0;

time\_t startTime = time(0);

for (int i = 0; i < 4; i++)

{

//Stopwatch starts

start[i] = clock(); //time calculation starts

// questions will be called here

for (int s = 0; s < 5; s++)

{

d.print\_quest();

cout << endl;

store\_ans(c);

system("cls");

c++;

time\_t currentTime = time(0); //stopwatch ends

int elapsedTime = difftime(currentTime, startTime);

int remainingTime = TIME\_LIMIT - elapsedTime;

if (remainingTime < 0) {remainingTime = 0; break; }

cout << "Time Remaining: " << remainingTime << " seconds" << endl;

this\_thread::sleep\_for(chrono::seconds(1)); // Wait for 1 second

remainingTime--;

}

stop[i] = clock(); //time calculation end

section\_wise[i] = static\_cast<float>(stop[i] - start[i]) / CLOCKS\_PER\_SEC;

cout << "Time taken for the "<<domain[i]<<" section: " << section\_wise[i] << " sec\n\n";

total\_time += section\_wise[i];

}

28

cout << "\t\t\tTotal time taken: " << total\_time << " sec" << endl;



}

};

class Calculate : public User

{

protected:

int score[20], sect\_mrks[4]= {0};

int total = 0;

double accuracy;

string indicator;

public:

void review() // question,user answer,correct answer,next option

{

//cout<<input[1]<<input[3];

for (int i = 0; i < 20; i++)

{

print\_quest();

while (n1 != 0);

cout << "Your answer: " << input[i] << endl;

cout << "Correct answer: " << ans[i] << endl;

cout << "Explanation: \n";

print\_method(i);

cout<<endl;

}

}

int check() // compare input[] with ans[]

{

int i = 0;

for (i; i < 20; i++)

{

if (input[i] == ans[i])

{

score[i] = 1;

total = total + 1;

}

}

return total;

}

void analysis() // prints total time,section\_time,section\_mrks,accuracy,

{

29

for (int i = 0; i < 20; i++)



{

if (score[i] == 1)

{

accuracy += 1;

}

}

accuracy = accuracy \* 5; // (accuracy/20)\*100

cout << "Your accuracy of correct answers: " << accuracy << " %" << endl<<endl;

int i=1;

for (int n = 0; n < 4; n++)

{

float i2=1;

do

{

sect\_mrks[n] = sect\_mrks[n] + score[i-1];

i2 = i % 5;

i++;

}

while(i2!=0);

cout<<"Marks in "<<domain[n]<<" section : "<<sect\_mrks[n]<<endl;

}

}

void strength\_weak() // based on sect\_mrks[4]assign strength or weakness //

// indicator

{

cout<<"This are your strength and weaknesses for each section\n";

for (int n = 0; n < 4; n++)

{

if (sect\_mrks[n] == 4 || sect\_mrks[n] == 5)

cout <<domain[n]<<":\nThis domain is your Strength. Make sure to utilize this in your career " << endl;

else if (sect\_mrks[n] == 3)

cout <<domain[n]<<":\nGood, work on improving your skills in this domain" << endl;

else if (sect\_mrks[n] == 1 || sect\_mrks[n] == 2)

cout <<domain[n]<<":\nThis domain is your Weakness. You need to work harder." << endl;

else

cout <<domain[n]<<":\nVery poor performance" << endl;

}

}

};

class Time\_Analysis : public Timer // Time based calculation

{

public:

void section\_time() // section wise time taken

{

30



cout<<"Your speed of answering the domains\n";

for (int i = 0; i < 4; i++)

cout<<domain[i]<<" domain: "<<section\_wise[i]<<endl;

}

};

int main()

{

database data;

User use;

User use1;

Calculate cal;

Time\_Analysis tim;

Timer t;

int i = 1;

while (i == 1)

{

if (data.instructions() == "ready")

{

cout<<"The test will begin now\n\n";

t.print\_time();

i = 0;

}

else

i = 1;

}

cout<<"\n\t\t\tYou have scored: "<<cal.check()<<"/20"<<endl;

string var;

cout << "\t\t\tCongratulations!! You have finished the test"<<endl<<endl;

do

{

cout<<"enter one of the option numbers to analyze or review your answers"<< endl;

cout << "1. Review" << endl;

cout << "2. Analysis" << endl;

cout << "3. Exit" << endl;

bool val = true;

while(val)

{

cin >> var;

if(var=="1"||var=="2"||var=="3")

{

val=false;

}

else

cout<<"Your answer will not be accepted. Enter valid options: ";

}

31



if(var=="1")

cal.review();

else if(var=="2")

{

cal.analysis();

cout<<endl<<endl;

tim.section\_time();

cout<<endl<<endl;

cal.strength\_weak();

cout<<endl<<endl;

}

else

cout << "Ending Analysis. Thank you for using our program." << endl;

}

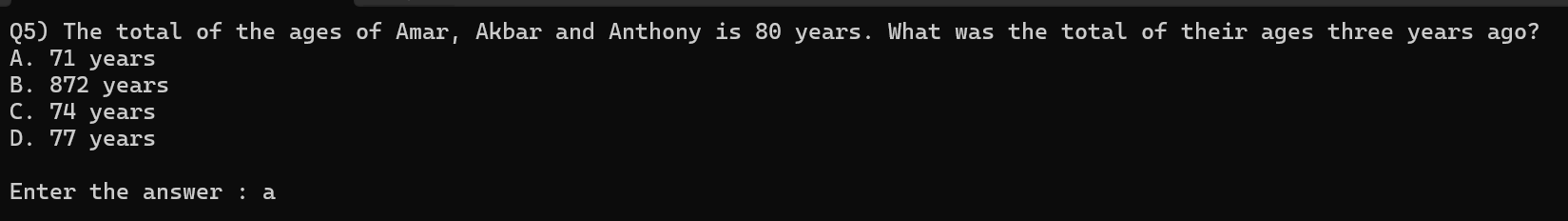
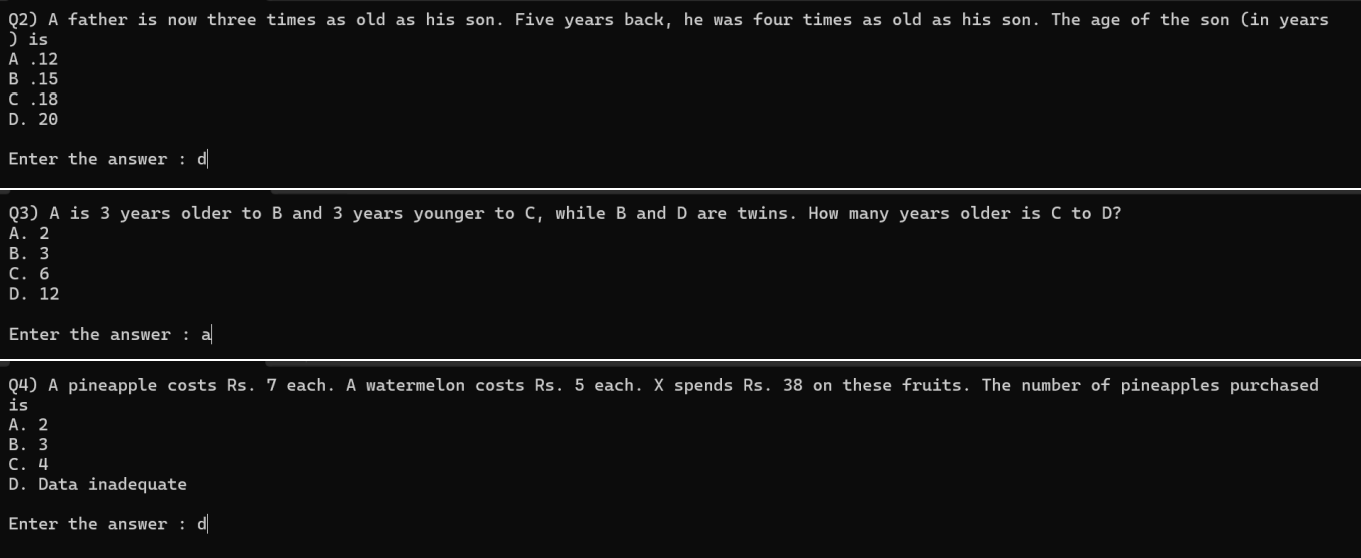
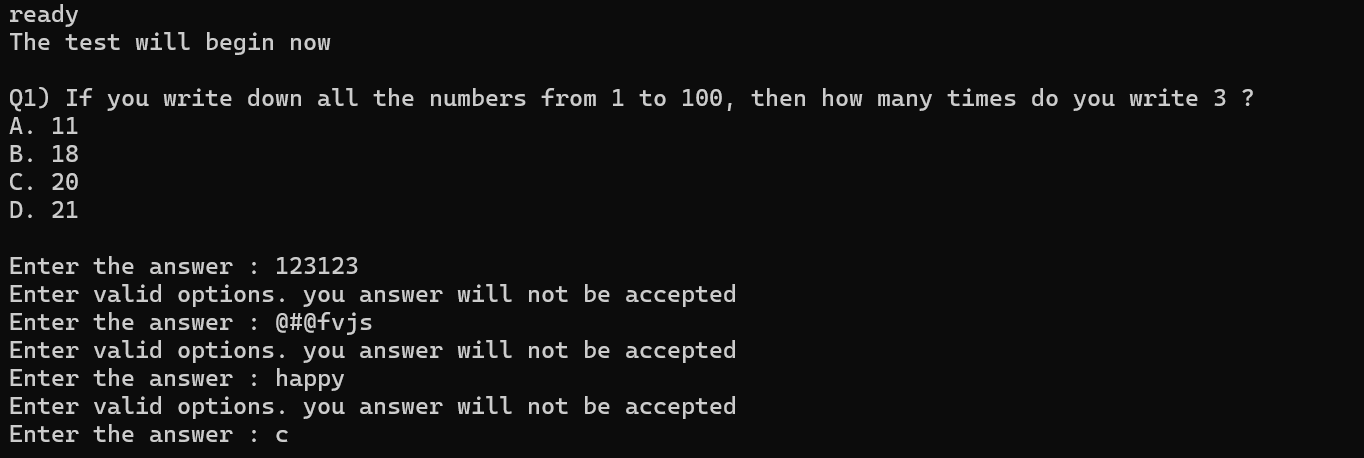
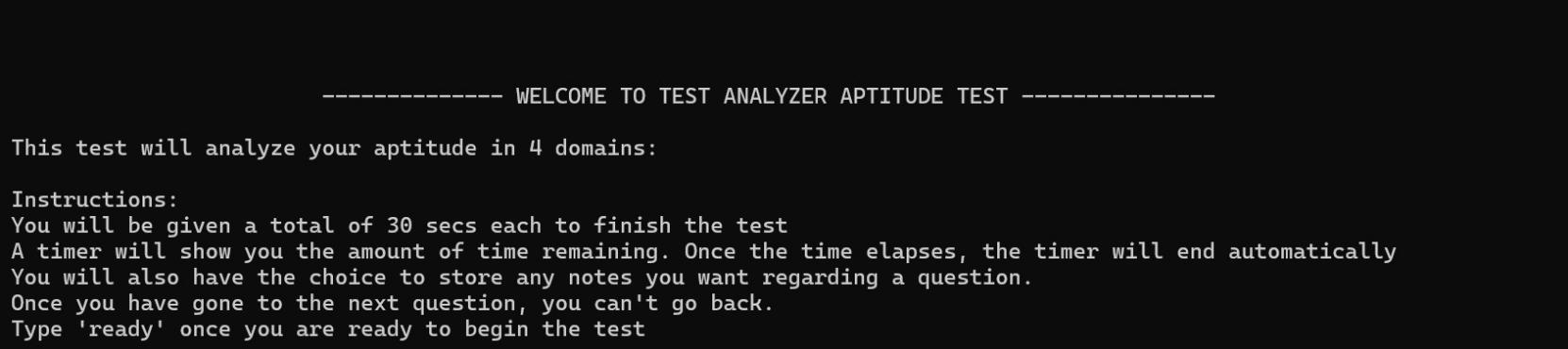
while(var!="3");

return 0;

}

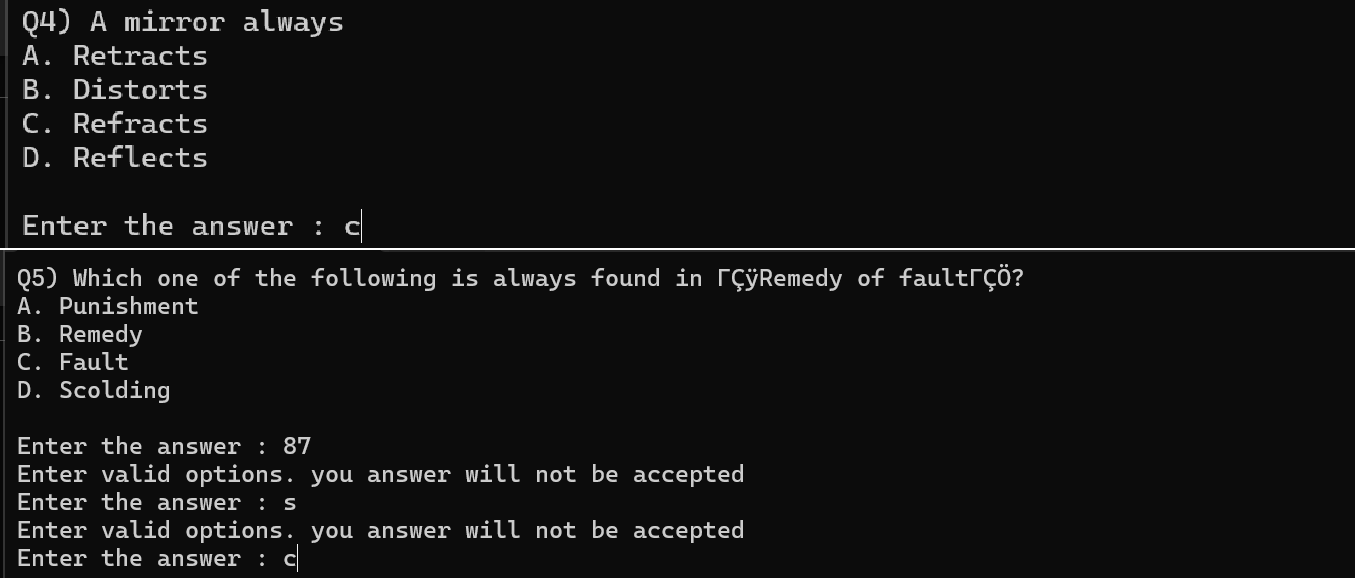
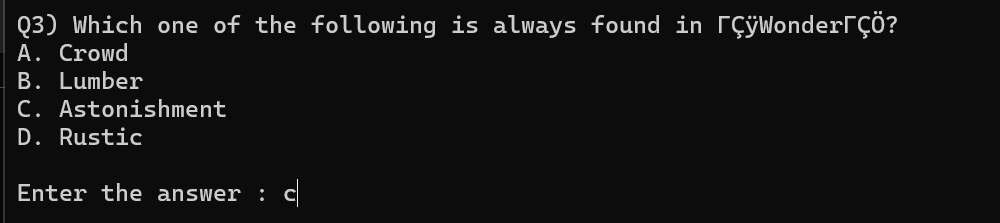
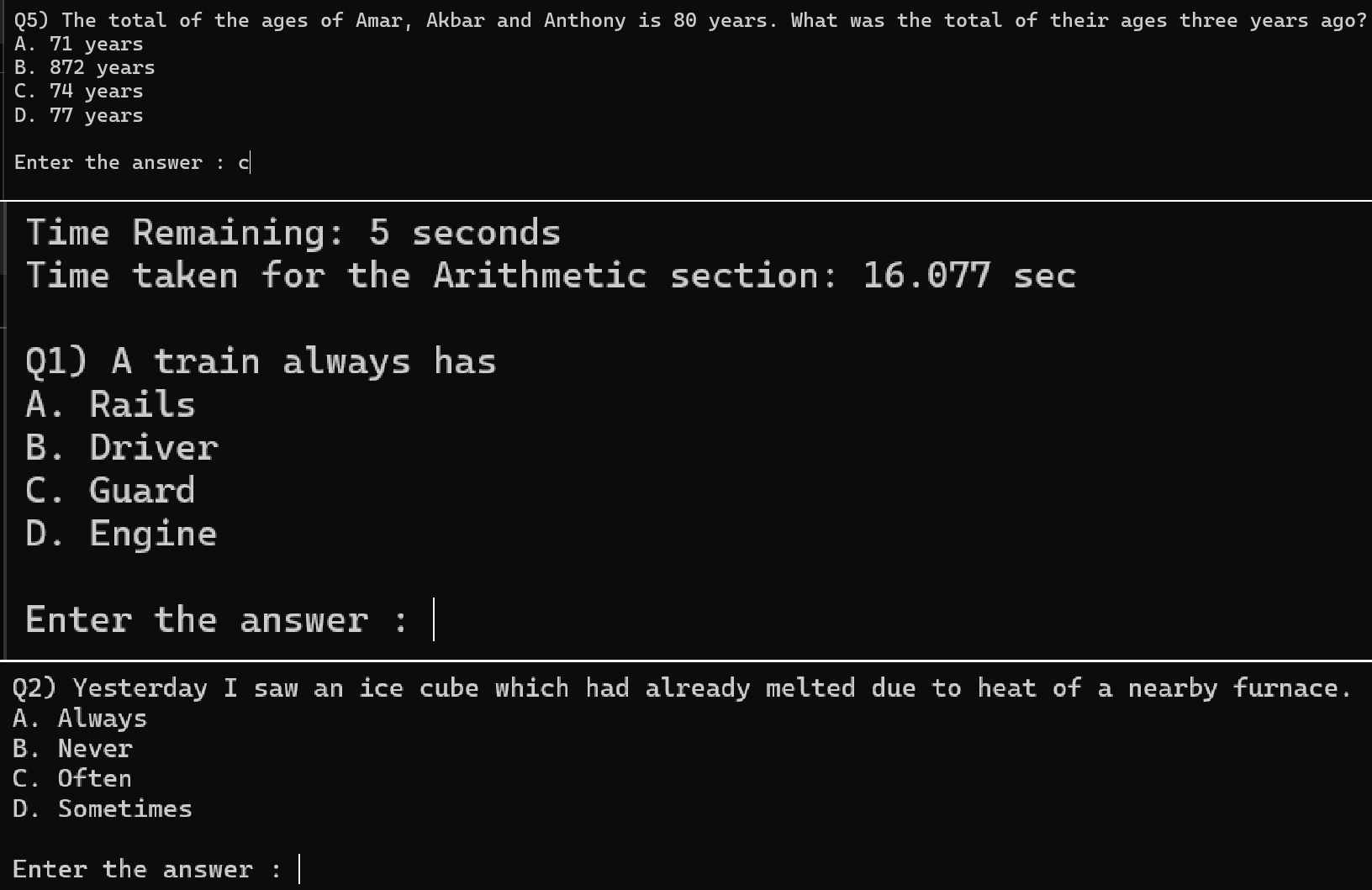
32

**17.**



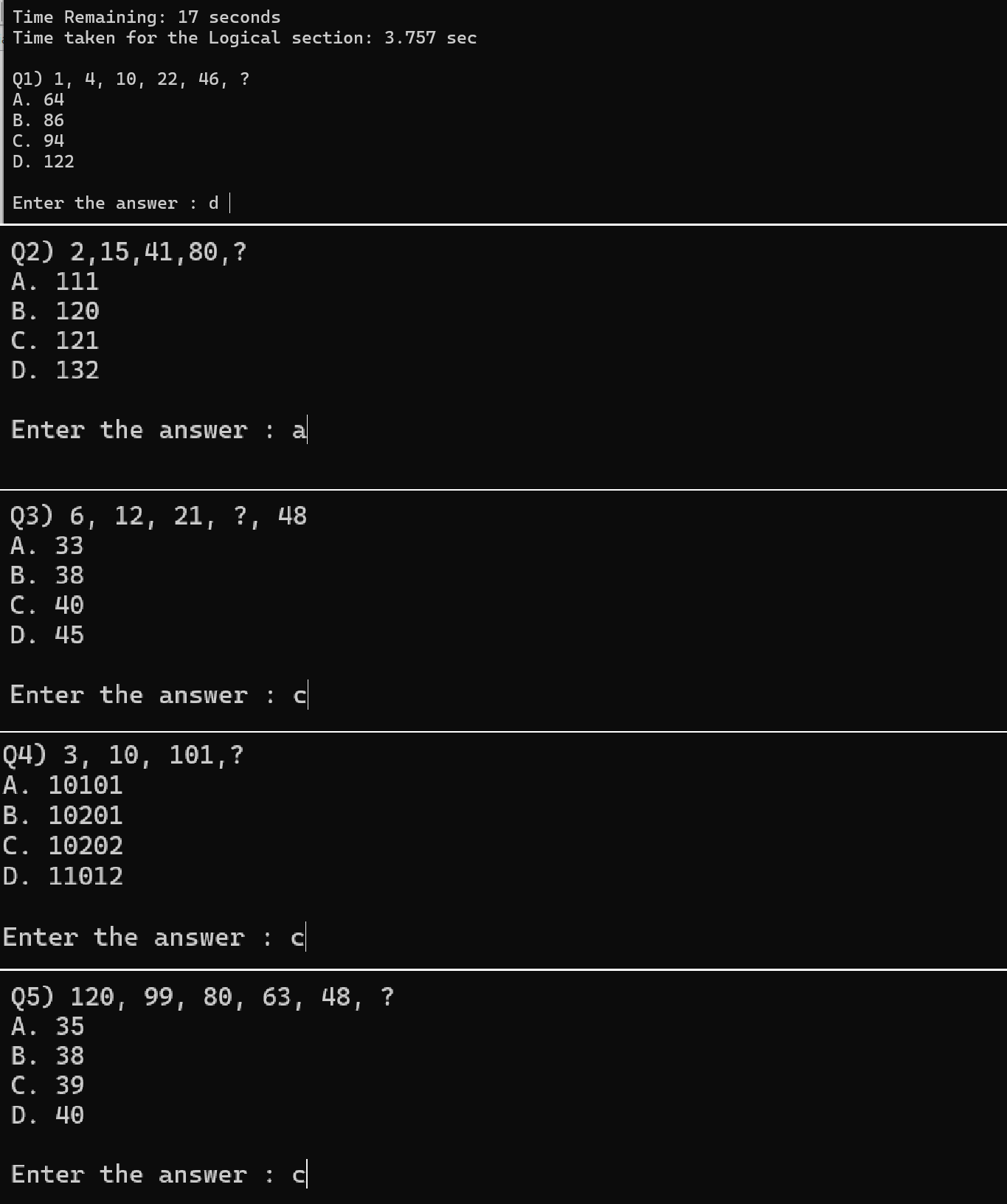
# Output screenshot

33

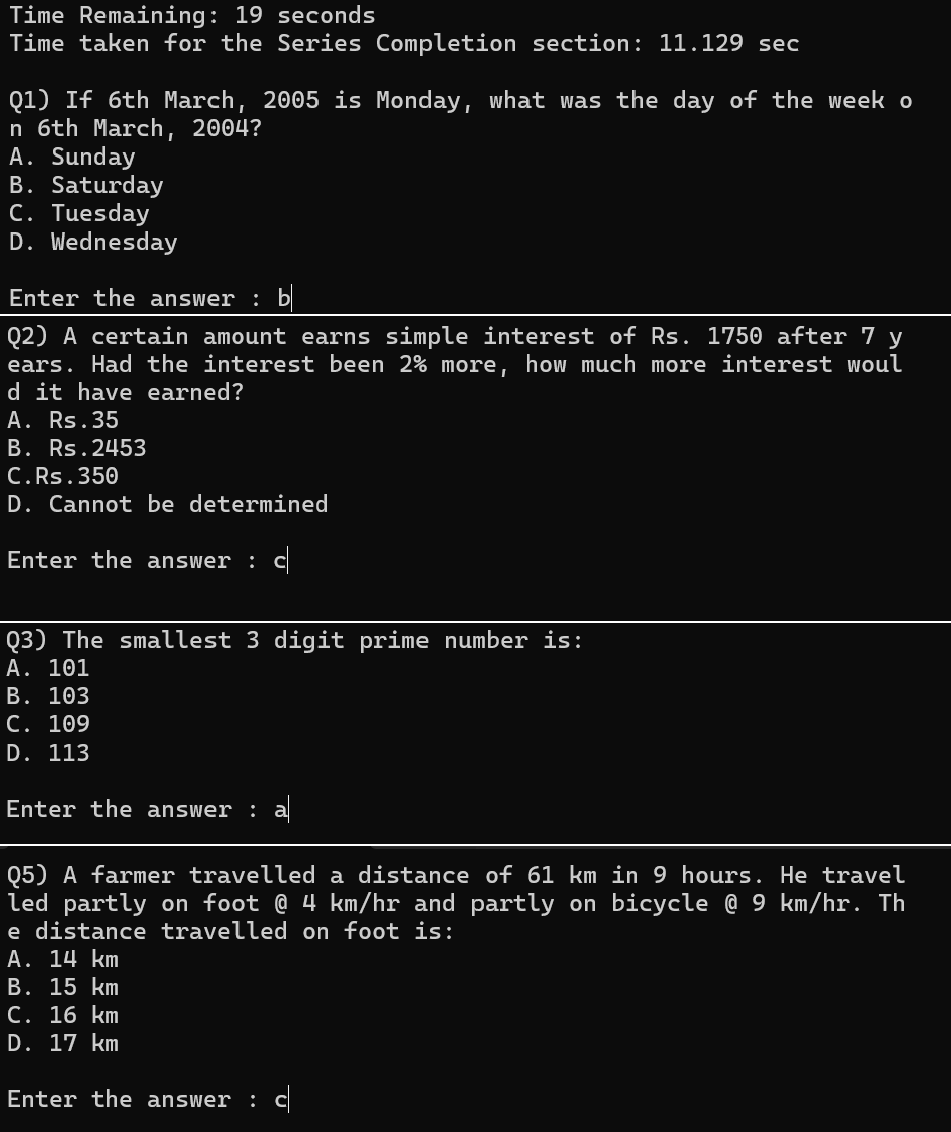


**(**this diagram shows how the program will respond if the user enters anything other than the option key in the answer)

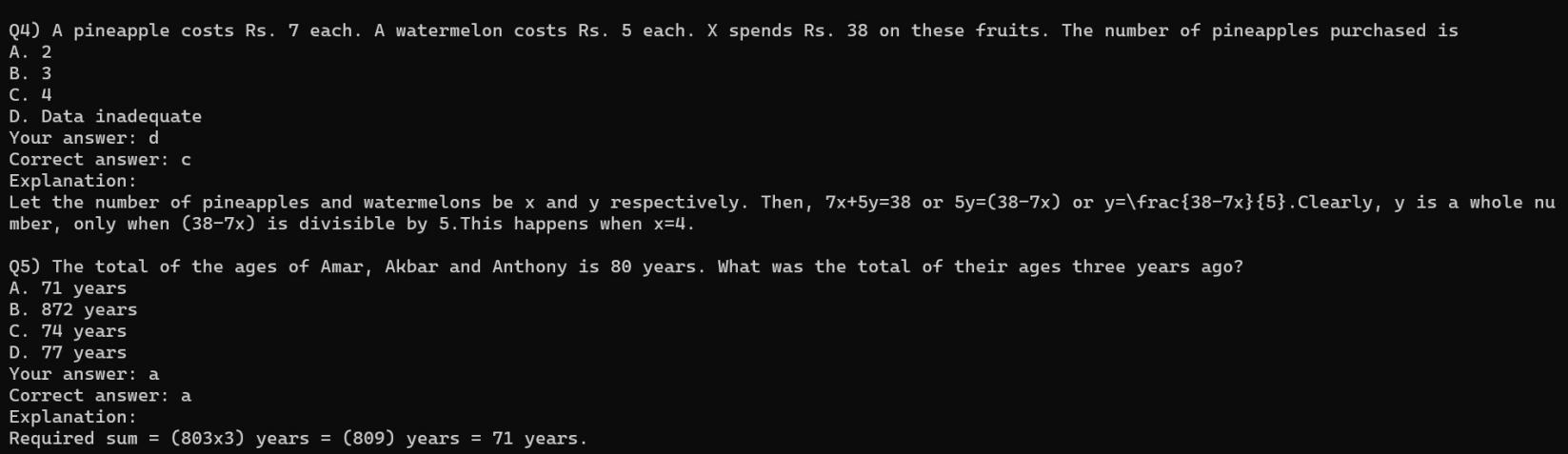
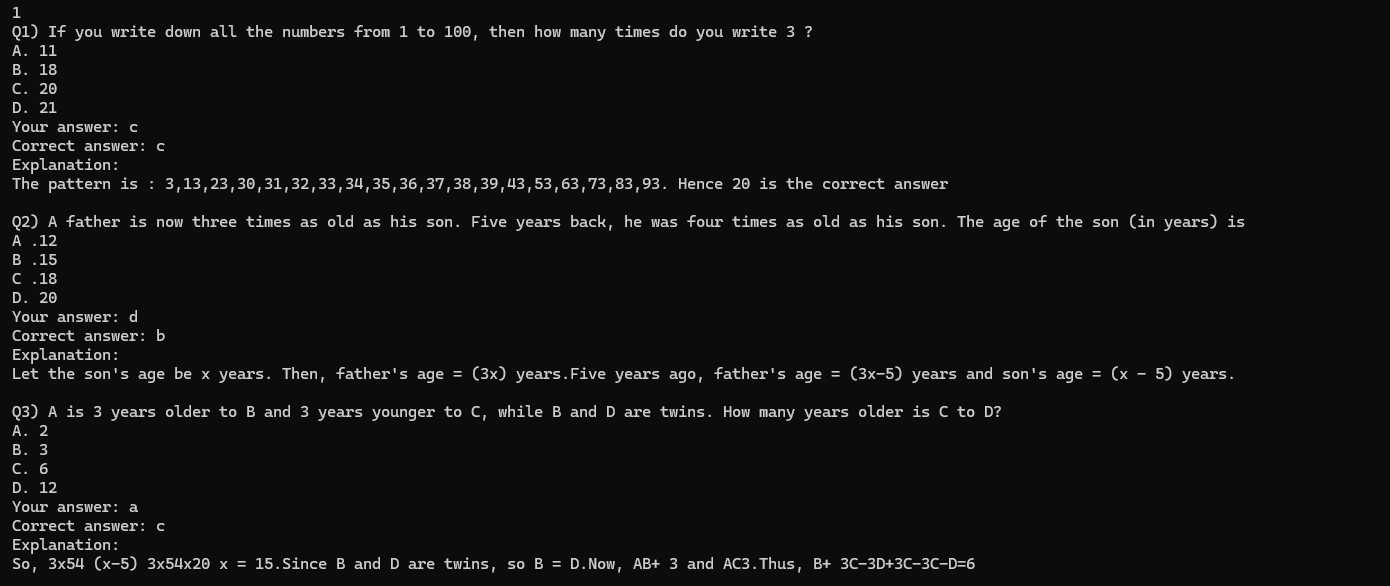
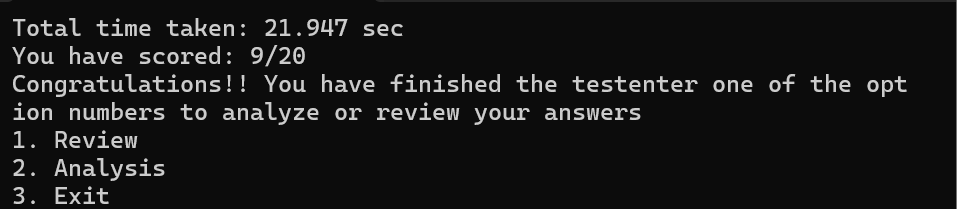
34



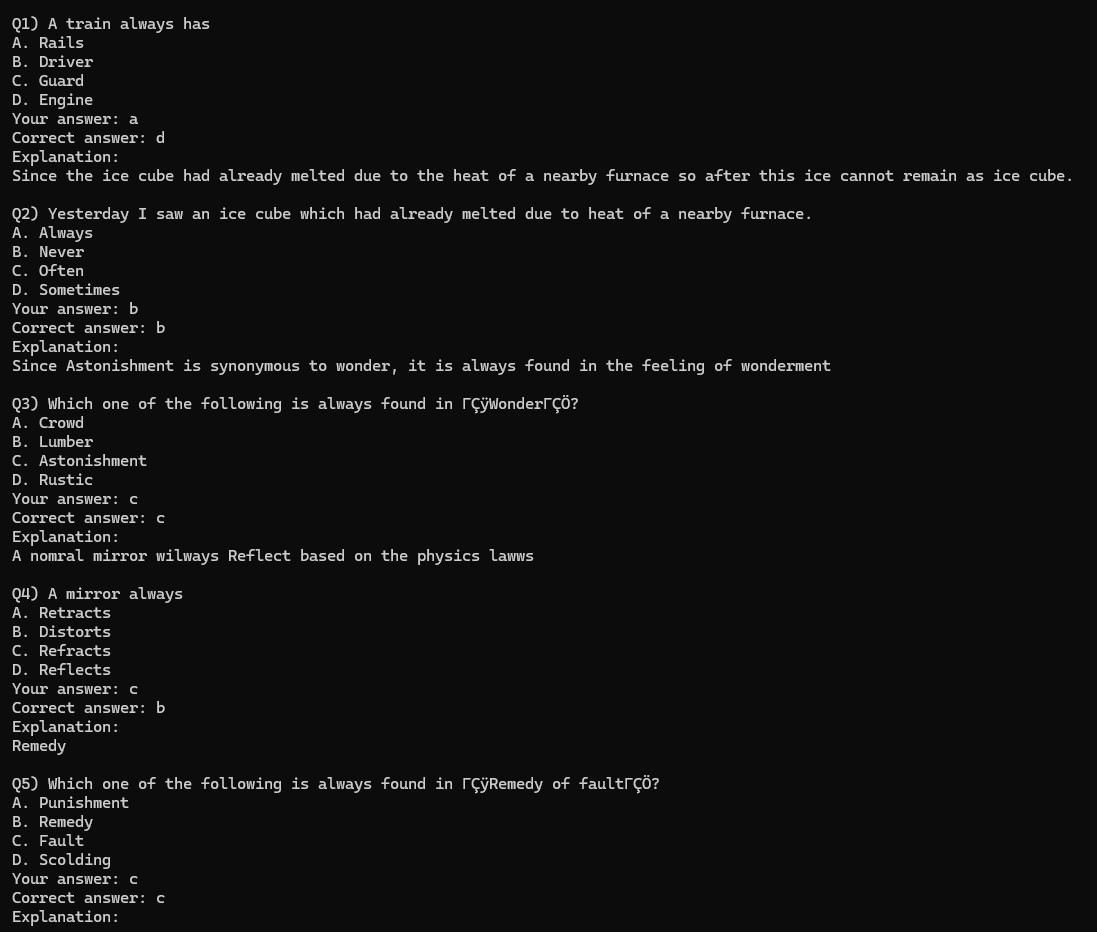
35



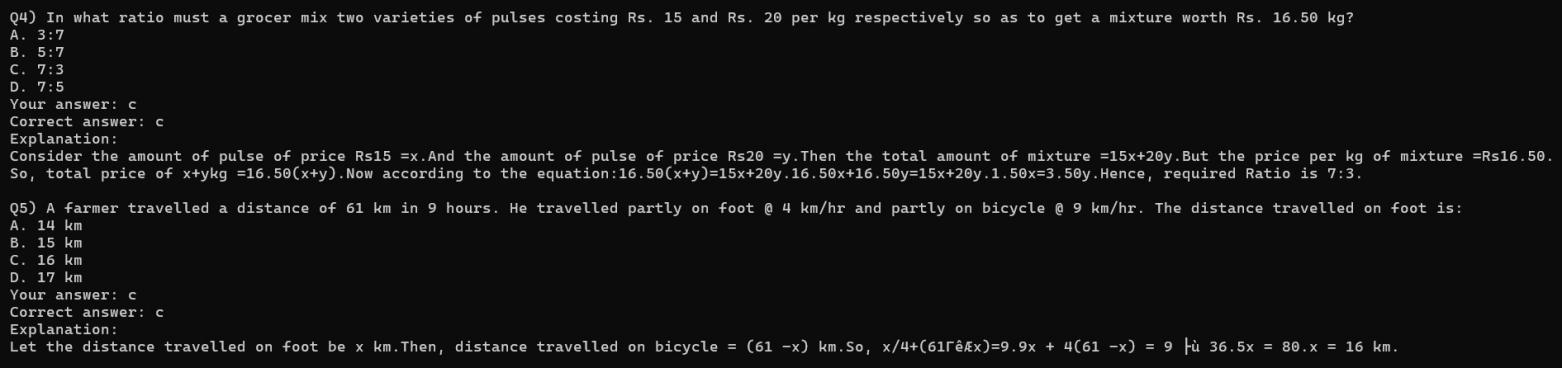
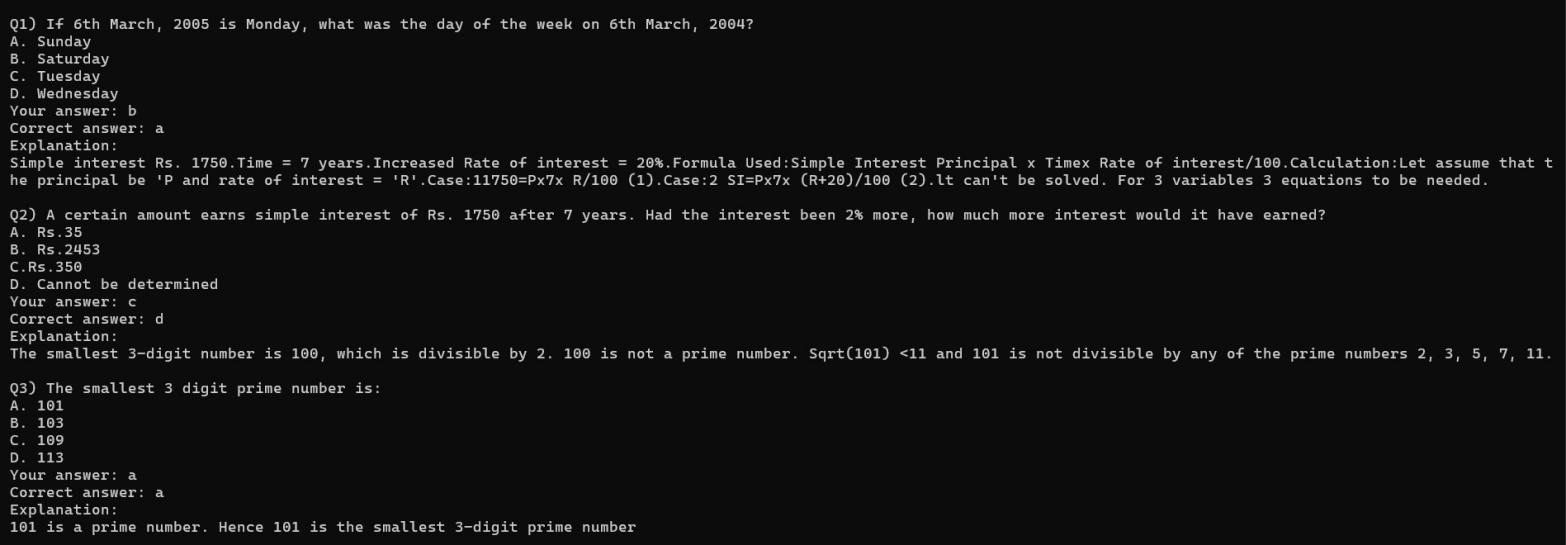
36



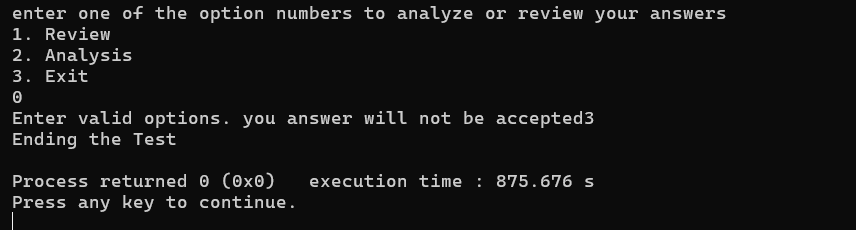
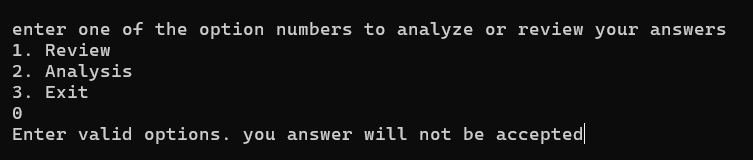
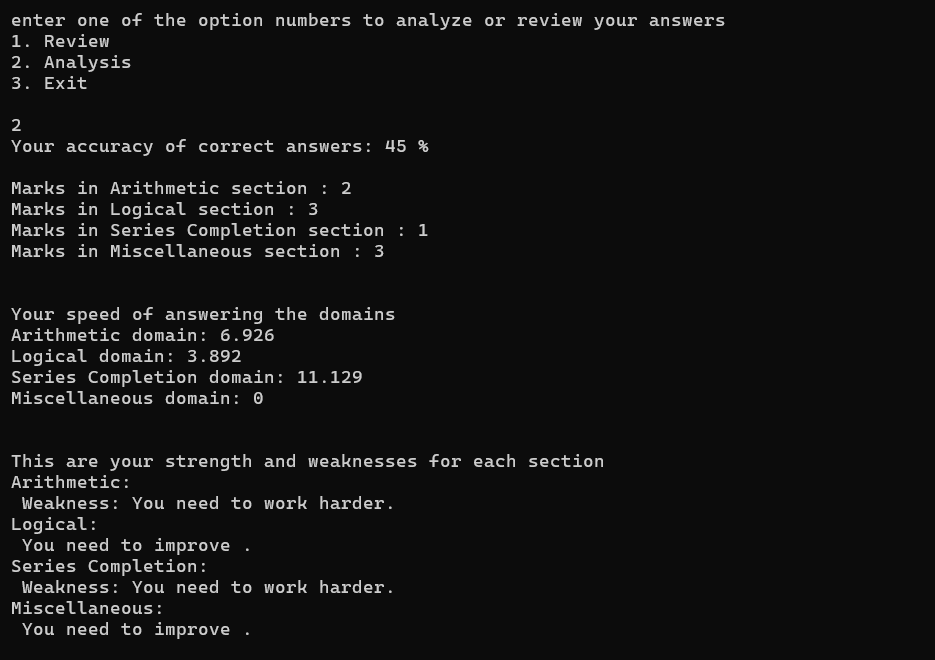
37



38



39



(this is the output tht will be printed if user enters an option number that dos not exist)

40

**18.**



# List of errors obtained and its solutions

* 1. **Preface:** Both The UML diagram and the code had a lot of errors . The errors in the UML diagram caused confusion in creating members of each class as extra or less members got created. There was also a mistake in depicting the inheritance from class to class, which caused an error in the code too. The errors in the code were numerous. These errors lead to output not being generated when data was called from txt files. Garbage value was often printed because of calculations. Inheritance was unsuccessful and the child class stored an empty copy of the variables instead storing the pre-existing data from the parent class
  2. Errors
     1. Our use case diagrams lacked use cases, so we included them.
     2. In the class our timer was not connected to the database. We modified the diagram.
     3. Our sequence diagram did not have proper sync with other diagrams which have now been modified and approved also, the names of the functionality.
     4. The error is that the file was opened in a different function than where it was required. Specifically, the file was opened outside the print\_quest() function but needed to be opened within that function to correctly print the questions from the database file. To resolve the error, the file should be opened within the print\_quest() function instead of opening it elsewhere. This ensures that the file is accessed at the appropriate time and location in the code. By opening the file within the print\_quest() function, you ensure that it is available for reading when needed to print the questions.
     5. Another error encountered is that our program is not able to print questions after the user has entered notes. This error was resolved by properly inheriting and calling print function responsible for printing the questions
     6. When the user enters an option other than a,b,c,d the program should prompt them to enter a valid option instead of entering an infinite loop. We have removed the problem by creating the variable using string data type. However, there is an unresolved error in which the string variables are not able to input strings there are sentences. Due to this the error statement loops as many times as the number of spaces entered in a string when user tries to enter a sentence as an answer instead of a word
     7. The program has been printing incorrect average speed. Later we realized the feature was not insightful to a user. For this purpose, we had to remove the feature from the code.

41

* 1. Conclusion:

A few features like user notes and continuous timer have been removed as they were showing errors or seemed unnecessary in hindsight after testing the code multiple times. Apart from that, all the errors are resolved and only a few limitations remain, which will be discussed in the next chapter.



**19.**

# Scope of improvement

* 1. preface:

Despite out best efforts in trying to include many concepts and features, a lot of have been excluded due to time constraints for the team, lack of complete knowledge and the errors that rose due to those features. Below are some of the features we believe would improve the code

* 1. Improvements
     1. To our knowledge, it is not possible, but a Graphical representation of the analysis section would make it easier for users to comprehend and analyze their performance and make the report less verbose and more diagrammatic.
     2. Many of the string Variables are not able to input strings with spaces between words, this must be improved with character array or vectors.
     3. The timer could have been for the entire program, instead of it showing only the time taken per section.

**19.2 Conclusion:**

Our team hopes to soon be able to optimize our analyzer program by including all possible features into the code to make it even more advanced and user friendly.

# References

1. E.Balaguruswamy, Object Oriented Programming in c++, 4 ed., 2008.
2. E. Balaguruswamy, Object Oriented Programming with C++, 4 ed., vol. 4th Edition, New Delhi: Tata McGraw Hill, 2008, p. 99.
3. R. C. Seacord, Secure Coding in C and C++, 2 ed., 2013.

42

##### 1% Overall Similarity

###### Top sources found in the following databases:

###### 0% Internet database 0% Publications database

###### Crossref database Crossref Posted Content database 1% Submitted Works database

**TOP SOURCES**

**The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.**

##### Grimsby College, South Humberside on 2013-05-21

[**1**](#_bookmark16)

**Submitted works**

## <1%

##### [philos.nmu.edu](http://philos.nmu.edu/cs228-01-22w/sample/parentchild/parent.cc)

[**2**](#_bookmark19)

**Internet**

<1%

##### Capella University on 2023-09-04

[**3**](#_bookmark14)

**Submitted works**

<1%

##### Trinity College Dublin on 2005-06-30 <1%

[**4**](#_bookmark12)

**Submitted works**

##### [coursehero.com](https://www.coursehero.com/file/225846919/HW03java/)

[**5**](#_bookmark20)

**Internet**

<1%

**Similarity Report**

**Sources overview**

**Similarity Report**

##### Excluded from Similarity Report

###### Bibliographic material Quoted material

###### Cited material Small Matches (Less then 10 words) Manually excluded text blocks

**EXCLUDED TEXT BLOCKS**

##### B Tech Computer Science and Engineering (Data Science)-311 Program

**engineering.nmims.edu**

##### SVKM’s NMIMSMukesh Patel School of Technology Management and Engineering...

**Harish Verlekar, Kashyap Joshi. "Ant & bee inspired foraging swarm robots using computer vision", 2017 Int...**

##### User interaction:The program interacts with the user by displaying

**Capella University on 2023-09-04**

##### this\_thread::sleep\_for(chrono::seconds(1)); // Wait for 1 second

[**www.tutorialspoint.com**](http://www.tutorialspoint.com/)

**Excluded from Similarity Report**