Object-Oriented Programming (CT-260)

Semester Project: BrainQuest (Report)

Department of Computer Science and Information Technology

Batch: 2023   Semester: Spring 2024

Group Members:

Faizan Khurram (AI-23014)

Muhammad Umair Hasan (AI-23018)

Muhammad Maaz Ur Rehman (AI-23036)

Mirza Abdul Raffay Baig (AI-23033)

1. Problem Statement:

We already know that every individual belongs to different races, casts etc. Similarly, every individual is also physically and mentally associated with an ability of intelligence, that is called the "Intelligence Quotient", or, more commonly termed as, the IQ score of an individual.

In this case, suppose the user wants to determine their level of intelligence, or, more formally, the user wants to figure out their IQ score. For this exact purpose, we test the user with multiple questions belonging to several different categories, for instance, Mathematics, Geography etc. and then the combined score of those correct questions helps us formulate the IQ score for the user. It is to note that the IQ score is determined based on the score criteria that it falls between for every individual.

Therefore, the project "BrainQuest" very efficiently takes responsibility of this purpose and calculates the IQ score for the current user that is performing the test by allowing the user to test their mental knowledge over a variety of different questioning categories, based on their preference of playing.

2. Project Description:

As the name suggests, the project BrainQuest is literally a "quest" for the user to figure out their intelligence quotient.

The project first welcomes and initializes the user with respective username and password (the password is essential for logging out the user at the end in order to save the final statistics). The user then gets presented with a variety of menu options, some of which will be covered in the *Functionalities Section* later ahead in the report, but the most crucial things of this project are the playing categories, that help in calculating the IQ.

There are 4 playing categories that the user can wish to test their IQ in, Maths, Sports, History and Riddle. Each playing category consists of 4 difficulty levels, Easy, Medium, Hard and Random (Frenzy) Mode. All of these are well explained to the user during the game (menu option). There are 15 questions for each level of difficulty, so for instance, the entire Sports category will have 45 questions, making a total of 180 questions for all the 4 categories of playing.

The most important aspect for this project, in the eyes of OOP itself, are the classes and distribution. This project consists of a total of 7 classes, which are the following:

-Player Class: Handles all the necessary operations for the player.

-Question Class: Each question that is presented, is treated like an object, possessing features like the question itself, list of answers, correct answer etc. Each question for any category, belongs to this class. It efficiently handles and maintains all the question objects for the program. The Player class interacts with the question objects, forming a dependency relationship with this class.

-BrainQuest Class: Handles all the necessary operations of the program structure and program flow. For example, it consists of the menu, rules etc. along with the play function, that is the core foundation of this project. It interacts with objects of Player Class and objects of derived classes of the base Question class.

-Maths Class (derived publicly from Question): Holds all the questions of Maths category (a total of 45 including Easy, Medium and Hard).

-Sports Class (derived publicly from Question): Holds all the questions of Sports category.

-History Class (derived publicly from Question): Holds all the questions of History category.

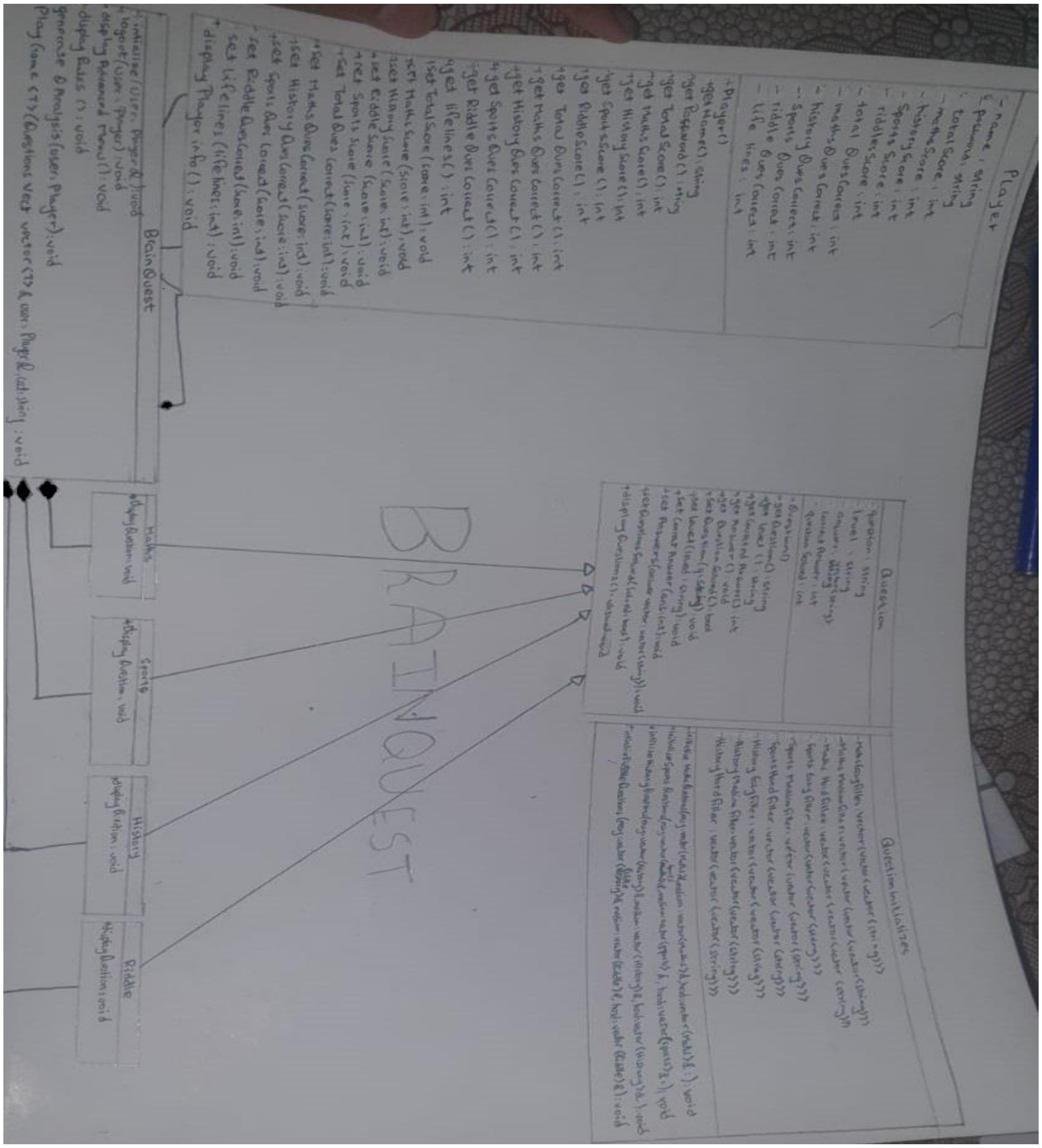
-Riddle Class (derived publicly from Question): Holds all the questions of Riddle category.

-QuestionInitializer Class: This class is used to bombard, or fill up the question objects (for each playing category) with the relevant questions (also keeping in mind the relevance to their level of difficulty). This class works on the back-end of the program and gets executed instantly as the program is executed. It does not have any relationship link with the other classes.

All the classes have been beautifully distributed to a total of 6 files, maintaining proper readability, understandability of the code, as well as the entire flow of the program and the program structure as well. These 6 files are automatically called as soon as the main function gets executed. Each class has been declared as a complete prototype first, followed by the implementations of all the necessary member functions.

These can be observed in the respective files in which the required class resides in.

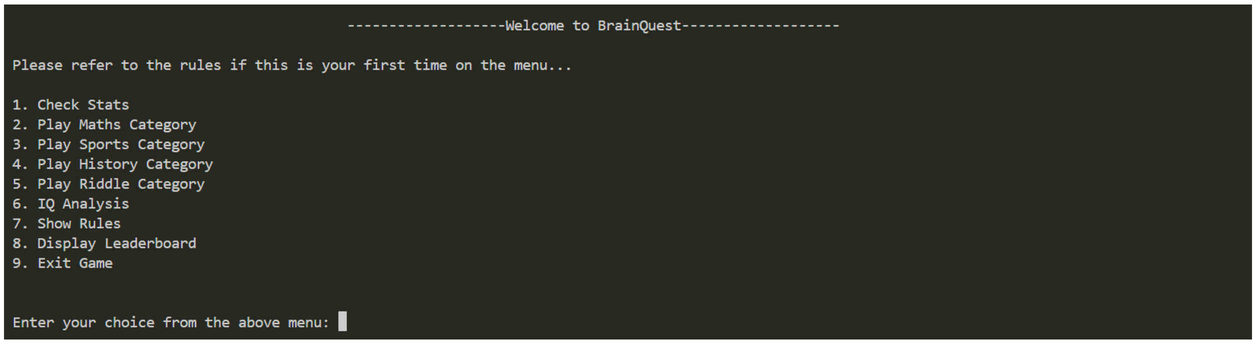
3. UML DIAGRAM:


Player
- name : string
- password : string
- totalScore : int
- mathsScore : int
- historyScore : int
- sportsScore : int
- riddlesScore : int
- total QuesCorrect : int
- mathsQuesCorrect : int
- historyQuesCorrect : int
- sportsQuesCorrect : int
- riddleQuesCorrect : int
- lifelines : int
+Player()
+getName() : string
+getPassword() : string
+getTotalScore() : int
+getMathsScore() : int
+getHistoryScore() : int
+getSportsScore() : int
+getRiddleScore() : int
+getTotalQuesCorrect() : int
+getMathsQuesCorrect() : int
+getHistoryQuesCorrect() : int
+getSportsQuesCorrect() : int
+getRiddleQuesCorrect() : int
+getLifelines() : int
+setTotalScore(score : int) : void
+setMathsScore(score : int) : void
+setHistoryScore(score : int) : void
+setRiddleScore(score : int) : void
+setSportsScore(score : int) : void
+setTotalQuesCorrect(score : int) : void
+setMathsQuesCorrect(score : int) : void
+setHistoryQuesCorrect(score : int) : void
+setSportsQuesCorrect(score : int) : void
+setRiddleQuesCorrect(score : int) : void
+setLifelines(lifelines : int) : void
+displayPlayerInfo() : void
BrainQuest
+initialize(user : Player&) : void
+logout(user : Player) : void
+displayAdvancedMenu() : void
+displayRules() : void
+generateQAnalysis(user : Player) : void
+PlayGame<T>(Questions : vector<T>& user : Player& cat : string) : void
Question
Maths
+displayQuestion : void
Sports
+displayQuestion : void
History
+displayQuestion : void
Riddle
+displayQuestion : void
QuestionInitializer
BRAINQUEST

4. Functionalities:

-Organized Menu-Driven Display:

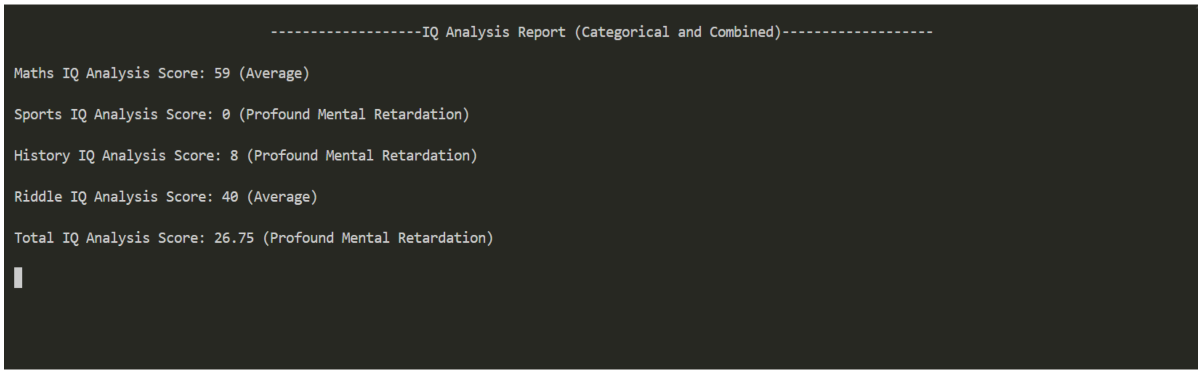
Created an organized and optimal menu display for the user, that serves as one of the most important displaying features to handle every single task of the user.



```
                                          ------------------Welcome to BrainQuest------------------

Please refer to the rules if this is your first time on the menu...

1. Check Stats
2. Play Maths Category
3. Play Sports Category
4. Play History Category
5. Play Riddle Category
6. IQ Analysis
7. Show Rules
8. Display Leaderboard
9. Exit Game


Enter your choice from the above menu:
```

-Generative IQ Analysis (Categorical and Combined):

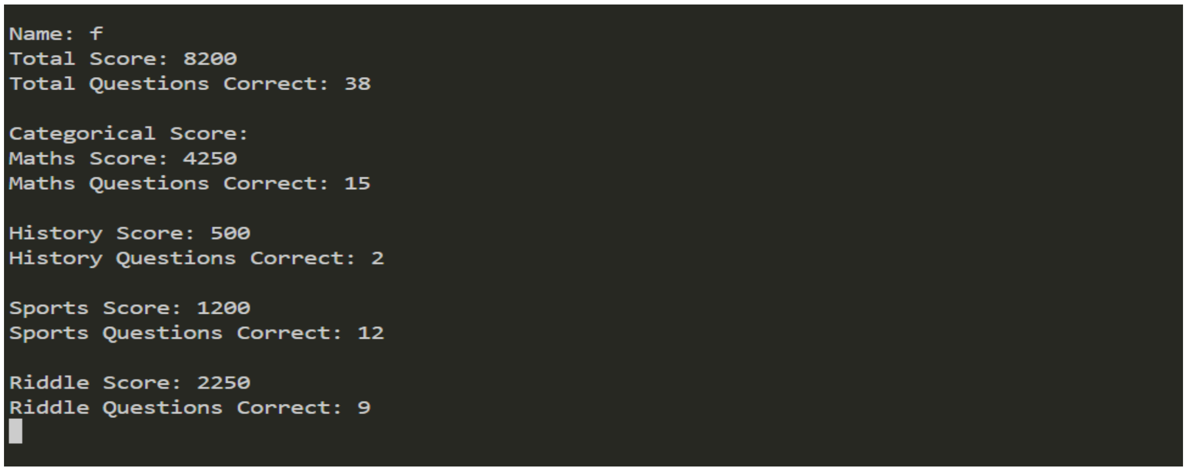
Implemented the main purpose of this program, the IQ Analysis Report. It is displayed to the user at will (whenever the user selects the respective menu option, it will generate the current IQ Report for the user) and when the user decides to exit (here, the final IQ Report will be displayed). ***The IQ Analysis is generated on the basis criteria of the total score obtained by the user for that particular category. The total combined IQ would be equal to the average of the IQ of the user in each respective category of playing*.**



```
                                ------------------IQ Analysis Report (Categorical and Combined)------------------

Maths IQ Analysis Score: 59 (Average)

Sports IQ Analysis Score: 0 (Profound Mental Retardation)

History IQ Analysis Score: 8 (Profound Mental Retardation)

Riddle IQ Analysis Score: 40 (Average)

Total IQ Analysis Score: 26.75 (Profound Mental Retardation)
```

-Statistics Display (Categorical and Combined):

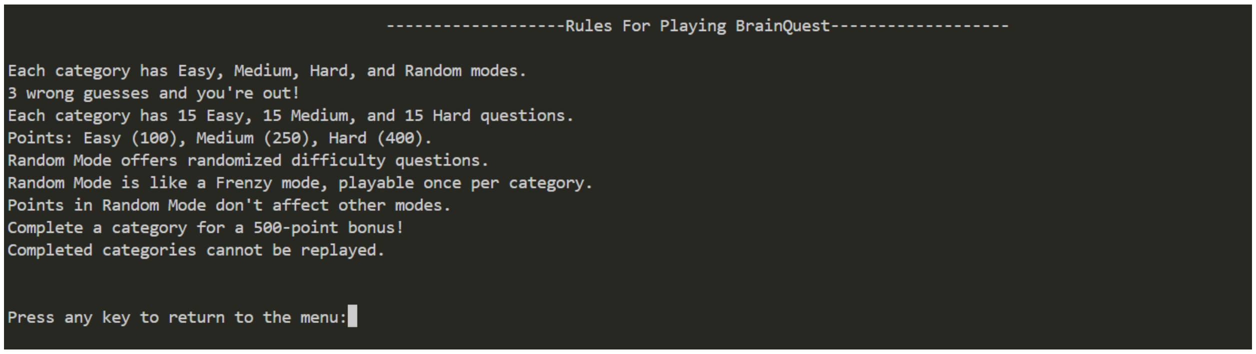
Implemented a well-organized function for displaying the statistics to the user. Just like the IQ Analysis Report, this is displayed to the user either by the user's will (through the menu options, in this case, the current statistics will be displayed) and when the user wishes to exit the program (in this case, the final statistics will be displayed)



```
Name: f
Total Score: 8200
Total Questions Correct: 38

Categorical Score:
Maths Score: 4250
Maths Questions Correct: 15

History Score: 500
History Questions Correct: 2

Sports Score: 1200
Sports Questions Correct: 12

Riddle Score: 2250
Riddle Questions Correct: 9
```

-BrainQuest Rules Display:

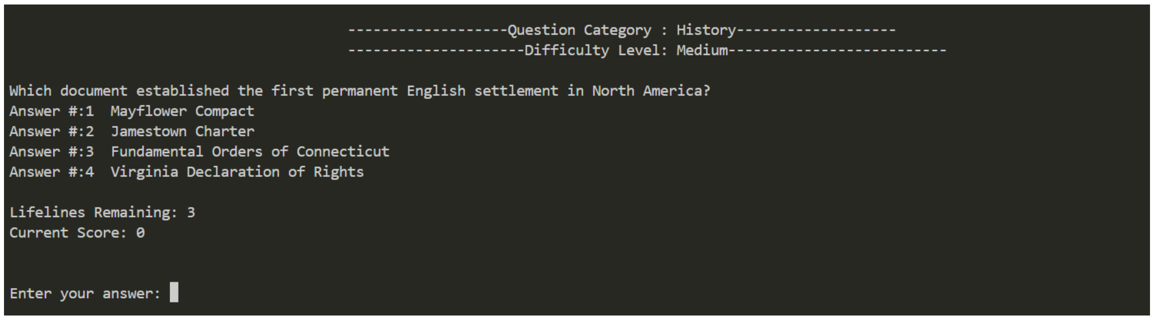
Implemented a function for displaying the Rules of BrainQuest. It neatly displays all the playing rules, making sure the user gets ready to test all the various categories and various functionalities presented before them on the main menu.



```
                                    ------------------Rules For Playing BrainQuest------------------

Each category has Easy, Medium, Hard, and Random modes.
3 wrong guesses and you're out!
Each category has 15 Easy, 15 Medium, and 15 Hard questions.
Points: Easy (100), Medium (250), Hard (400).
Random Mode offers randomized difficulty questions.
Random Mode is like a Frenzy mode, playable once per category.
Points in Random Mode don't affect other modes.
Complete a category for a 500-point bonus!
Completed categories cannot be replayed.


Press any key to return to the menu:
```

-Question Display and Terminal-Based Interface:

Implemented a properly formatted question displaying function for the user, that has the same implementation for any category that the user is playing in. The only difference is of the categorical display and difficulty level.



```
                                   -------------------Question Category : History-------------------
                                   --------------------Difficulty Level: Medium-------------------------

Which document established the first permanent English settlement in North America?
Answer #:1  Mayflower Compact
Answer #:2  Jamestown Charter
Answer #:3  Fundamental Orders of Connecticut
Answer #:4  Virginia Declaration of Rights

Lifelines Remaining: 3
Current Score: 0


Enter your answer:
```

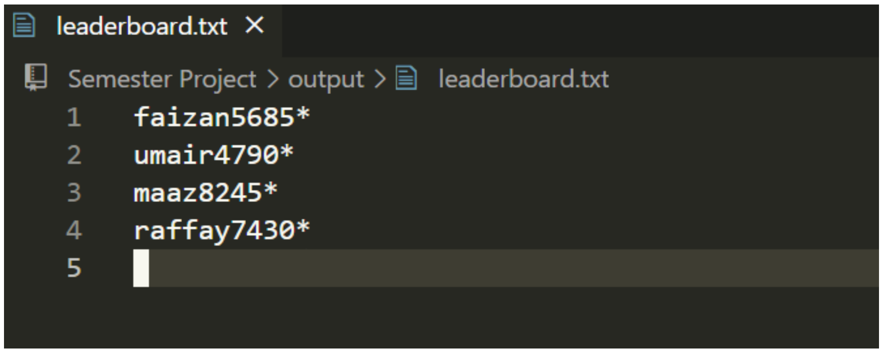
-CLS() and Sleep() Optimization:

For the most suitable and best performance in terms of readability and understandability, we have efficiently implemented the use of the system("cls") function from the "<windows.h>" header file, along with the sleep() function from the "<unistd.h>" header file. This allows the best flow of our program on the terminal, maintaining neat and clean flow of the code and best understandability for the user.

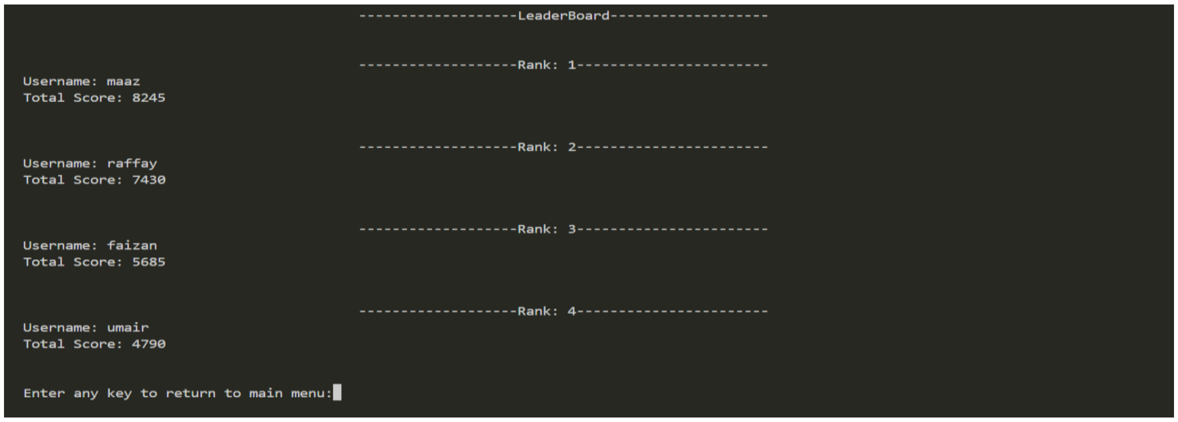
-Dynamically Updating and Self-Sorting Leaderboard:

One of the most beneficial, fancy and important corner-stone of this project is the dynamically updating and self-sorting leaderboard. It efficiently manages the records of all the users who have, in the past, played BrainQuest and, displays their data sorted from top-to-bottom **(in terms of the total score instead of the IQ score**) to the current user who has wished to see the current standings on the leaderboard.

When the user decides to finish the IQ test, their starts get automatically saved to a file "leaderboard.txt", that is operating at the back-end of this project. Once the stats get saved, the leaderboard automatically takes those stats and, based on the basis of the score of the user, self-sorts and positions that user at the correct place on the leaderboard. ***The format for writing the stats of the user on the .txt file have been defined on custom logic basis*. *It is necessary for the user to terminate the program through the "exit" menu option. Abrupt ending or killing the terminal will not save the stats on the leaderboard.txt file*.**



```
faizan5685*
umair4790*
maaz8245*
raffay7430*
```

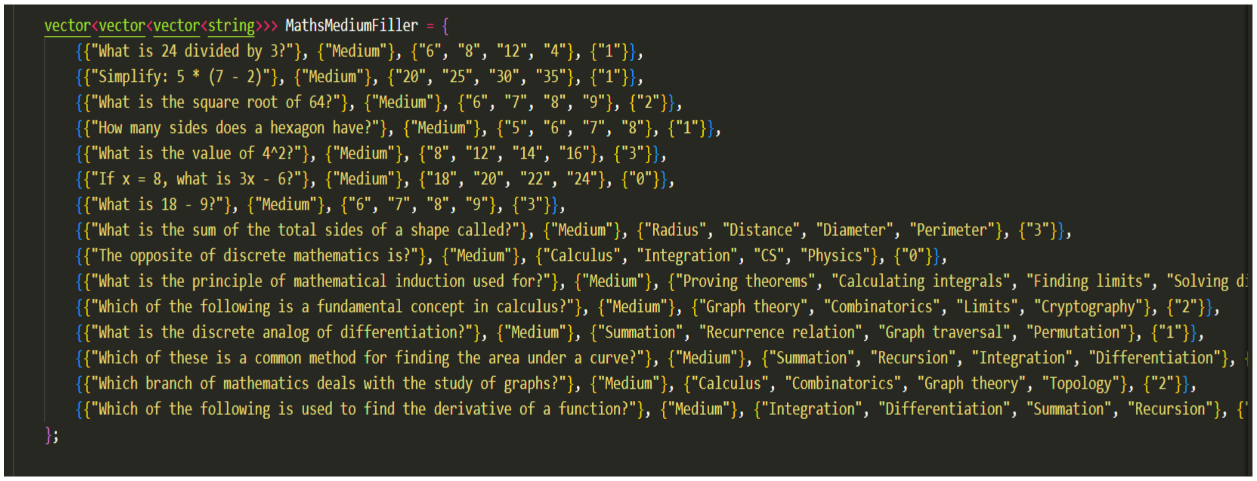


```
                                        ------------------LeaderBoard------------------

                                        ------------------Rank: 1----------------------
Username: maaz
Total Score: 8245


                                        ------------------Rank: 2----------------------
Username: raffay
Total Score: 7430


                                        ------------------Rank: 3----------------------
Username: faizan
Total Score: 5685


                                        ------------------Rank: 4----------------------
Username: umair
Total Score: 4790

Enter any key to return to main menu:
```

-Back-End Question-Initializer Class:

Implemented an isolated and separate class named “QuestionInitializer” that also serves as another important corner-stone of this project. The main obstacle was figuring out the loading scheme of all the collected questions into their respective question objects, which lead to the logic of implementing such type of a class that performs all of this instantly on the back-end, as soon as the main function gets executed. ***The entire logic of this question objects bombarding has been done completely on custom basis*.**

One example prototype of this class is the following vector that holds the questions for Maths Medium category:



```
vector<vector<vector<string>>> MathsMediumFiller = {
    {{"What is 24 divided by 3?"}, {"Medium"}, {"6", "8", "12", "4"}, {"1"}},
    {{"Simplify: 5 * (7 - 2)"}, {"Medium"}, {"20", "25", "30", "35"}, {"1"}},
    {{"What is the square root of 64?"}, {"Medium"}, {"6", "7", "8", "9"}, {"2"}},
    {{"How many sides does a hexagon have?"}, {"Medium"}, {"5", "6", "7", "8"}, {"1"}},
    {{"What is the value of 4^2?"}, {"Medium"}, {"8", "12", "14", "16"}, {"3"}},
    {{"If x = 8, what is 3x - 6?"}, {"Medium"}, {"18", "20", "22", "24"}, {"0"}},
    {{"What is 18 - 9?"}, {"Medium"}, {"6", "7", "8", "9"}, {"3"}},
    {{"What is the sum of the total sides of a shape called?"}, {"Medium"}, {"Radius", "Distance", "Diameter", "Perimeter"}, {"3"}},
    {{"The opposite of discrete mathematics is?"}, {"Medium"}, {"Calculus", "Integration", "CS", "Physics"}, {"0"}},
    {{"What is the principle of mathematical induction used for?"}, {"Medium"}, {"Proving theorems", "Calculating integrals", "Finding limits", "Solving di
    {{"Which of the following is a fundamental concept in calculus?"}, {"Medium"}, {"Graph theory", "Combinatorics", "Limits", "Cryptography"}, {"2"}},
    {{"What is the discrete analog of differentiation?"}, {"Medium"}, {"Summation", "Recurrence relation", "Graph traversal", "Permutation"}, {"1"}},
    {{"Which of these is a common method for finding the area under a curve?"}, {"Medium"}, {"Summation", "Recursion", "Integration", "Differentiation"}, {
    {{"Which branch of mathematics deals with the study of graphs?"}, {"Medium"}, {"Calculus", "Combinatorics", "Graph theory", "Topology"}, {"2"}},
    {{"Which of the following is used to find the derivative of a function?"}, {"Medium"}, {"Integration", "Differentiation", "Summation", "Recursion"}, {'
};
```

5. Principles of OOP Utilized:

-Encapsulation:

All the attributes instances are combined efficiently into a single unit (class) that promotes better understandability and re-usability of the code. All attributes and member functions have been **encapsulated** efficiently.

-Data Hiding:

The use of public and private access modifiers has been implemented in each of the 7 classes. This allows structured access control of attributes and objects. In this case, all the necessary getter and setter member functions have been defined for each class (if required). All the attributes have been kept as private and all the member functions have been kept as public for each class.

-Inheritance:

One of the most important principles of OOP, Inheritance, has been implemented between the Question class and it's 4 derived classes (Maths, Sports, History, Riddle).

Since these categorical classes were derived from the abstract Question class, they shared the same attributes and member functions of a simple object that is holding a question.

To be specific, the relationship between the Question class and the 4 categorical classes is exhibiting **Hierarchical Inheritance.** This is the straightforward is-a relationship.

Another relationship, the uses-a, or more formally, the dependency relationship, is established between the Player Class and the Question Class, since the Player class uses, or interacts with each object belong to the Question class.

-Polymorphism:

Another important backbone of OOP, polymorphism, has been implement using the Question class. The question class consists of one polymorphic function, that gets implemented by each of the 4 categorical classes.

-Abstraction:

To be specific, the Question class consists of a polymorphic function, but it is set to be a **pure virtual function**. In this case, no implementation of it is found in the base (Question) class, due to which all the derived classes have presented the implementation for this pure virtual function. **This makes the Question class an Abstract class.**

-The static and const Keywords:

The usage of static and const keywords has been implemented on the member functions of the classes, primarily on member functions of the Player class and the question class. Every function used for displaying purpose has been declared as a const function, for instance, displaying info of the user etc. Functions like displaying menu have been made as static const, since they are both used for displaying purpose and they also directly belong to the class.

-Template Functions:

The most important function, that serves as the heart and soul of the entire project, is the play function. Instead of creating multiple copies of this function for each individual categorical class, it has been implemented like a template function in the BrainQuest class. In this case, no matter which class the object or vector or any other parameter might belong to, the implementation is the same for each. This saved around 400 lines of code.

-Friend Class:

A problem arose during the usage of two member functions of the BrainQuest class, specifically the initialize and logout member functions for an object of the Player class. These member functions were trying to access the attributes of the Player object, by default implementation. In this case, the entire BrainQuest class has been declared as a “friend” class for the “Player” class, to automatically allow member functions of

BrainQuest class to directly handle the attributes of Player class objects. Although in some member functions of the BrainQuest class, getter and setter functions have also been used to handle Player class objects and objects of each individual categorical question class. But, in some cases, the need for declaring the BrainQuest as a friend for Player class was found necessary. ***It is to note that all the objects of each categorical question class have been handled using their respective getter and setter functions, therefore, there was no need to declare BrainQuest class as a friend to each individual categorical class*.**

6. Contributions of Group Members:

Faizan Khurram:

-Implemented the dynamically updating and self-sorting leaderboard logic.

-Implemented the use of file handling (for the dynamic leaderboard) along with the logic of a single gameplay template function.

-Created the entire structure for the QuestionInitializer class for loading up all the question objects and also made important modifications and code improvisations.

Maaz Rehman:

-Implemented proper organization of code into separate header files, including neat and clean class prototypes and separate yet clearly visible class implementations.

-Created the structures of all the classes by filling them with all the necessary attributes and member functions (which were first all explained and written out in the UML Diagram by Umair)

-Proposed the logic behind the relationship links between the concerned clases.

Raffay Baig:

-Structured the entire flow of the program in the main function

-Proper menu handling and menu display logic control in the main function

-Implemented proper logic control for player member functions and their usages, as-well-as for player attributes handling and player attributes modifications

Umair Hasan:

-Created the entire UML diagram structure, the foundation of the project, along with the classes distribution concept for Maaz to implement inside the code.

-Integrated proper terminal sleep() and cls() optimization inside the code for better and clean program flow on the terminal.

-Contribute to acts for code debugging and logic debugging, fixing many logical errors near project completion.

-Implemented a few minor adjustments and improvisations.

7. Limitations / Future Enhancements:

-The project is only single player per program execution, a multi-player project could have also been implemented, but for now the focus was on logic, distribution and implementation.

-Exception Handling could have neatly been implemented, but currently, not such need was felt during the entire phase of project testing, since we concluded on letting the user continue with some parts of the program, instead of ending abruptly.

-An entire separate class for loading up all the Question objects has been implanted, Although, this can be updated to replacing it with some dynamic file (database) that randomizes every question and even changes and replaces some questions from time to time. For now, the separate initializer class was felt sufficient.

-There was no need of implementing some complex levels of inheritance between the concerned classes. The job was being done with the already implemented classes.

-Implementing a time bomb system for fast user decisions during answering questions can be considered as some sort of improvisation and future enhancement. For the current project, we limited it to the basis of lifelines (3 strikes and out).