COMP315
Individual Project Documentation

Members	
1. Lorenzo Munisami	217037162
2. Dashen Govender	218031968

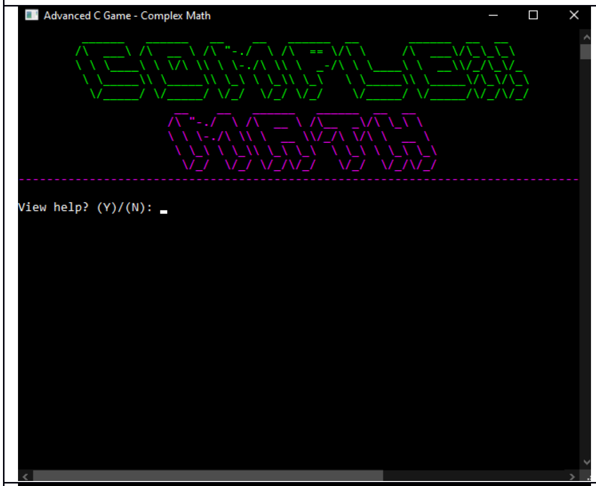
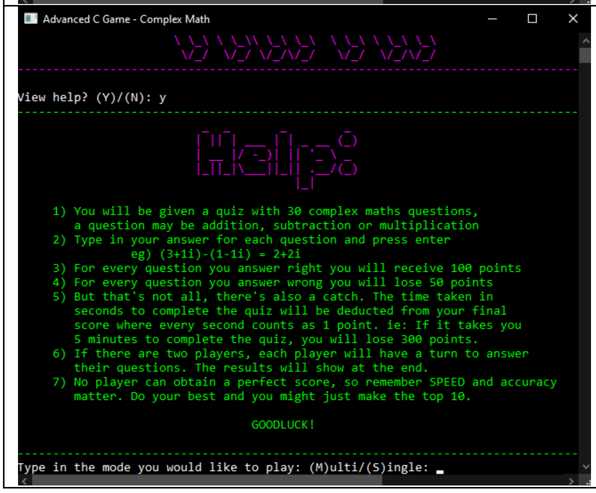
Contents

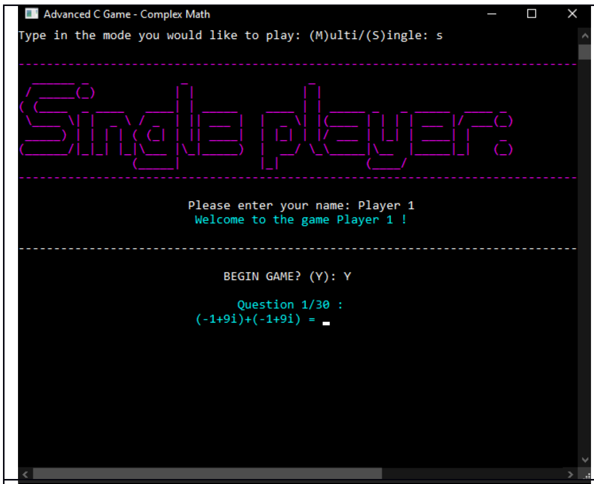
Introduction

Complex Math is a console-based quiz game used to introduce students ,in grades 10 to 12 who want to take Advanced Programme Mathematics (AP Maths) , to the basics of complex numbers and expansion of binomials. This quiz will help students strengthen their mathematical abilities that they have learnt in class in a fun and competitive manner.

The quiz comprises of a single player and a multiplayer mode with each player being asked 30 questions. The goal of the single player mode is to earn the highest score you can in as little time as possible. The score will be calculated according to the number of questions you get right (+100) and wrong (-50). The amount of time you take will also be deducted from your total score (1s = -1 point). In the multiplayer mode 2 players will face off with the winner being the player with the highest score earned. Each player will have a turn to answer their 30 questions. The game includes a Top 10 Leader board, a timer for each player, and help feature which will explain the rules of the game and which contains a basic introduction into complex numbers to help newcomers get a grip of the subject.

Screenshots

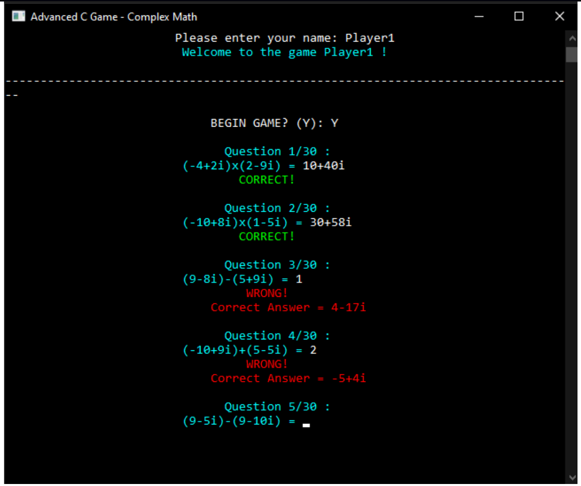
Screenshot	Explanation
 Advanced C Game - Complex Math View help? (Y)/(N):	**Welcome page:** Welcomes the player/s to the game and gives them an option to view the Help page.
 Advanced C Game - Complex Math View help? (Y)/(N): y 1) You will be given a quiz with 30 complex maths questions, a question may be addition, subtraction or multiplication 2) Type in your answer for each question and press enter eg) (3+1i)-(1-1i) = 2+2i 3) For every question you answer right you will receive 100 points 4) For every question you answer wrong you will lose 50 points 5) But that's not all, there's also a catch. The time taken in seconds to complete the quiz will be deducted from your final score where every second counts as 1 point. ie: If it takes you 5 minutes to complete the quiz, you will lose 300 points. 6) If there are two players, each player will have a turn to answer their questions. The results will show at the end. 7) No player can obtain a perfect score, so remember SPEED and accuracy matter. Do your best and you might just make the top 10. GOODLUCK! Type in the mode you would like to play: (M)ulti/(S)ingle:	**Help Page:** Displays the rules of the game



```
Type in the mode you would like to play: (M)ulti/(S)ingle: s
---------------------------------------------------------------------------------

                     Please enter your name: Player 1
                      Welcome to the game Player 1 !

---------------------------------------------------------------------------------

                          BEGIN GAME? (Y): Y

                            Question 1/30 :
                      (-1+9i)+(-1+9i) = 
```

Single player Mode select:

If the player/s select single player mode, they will need to fill in their name.

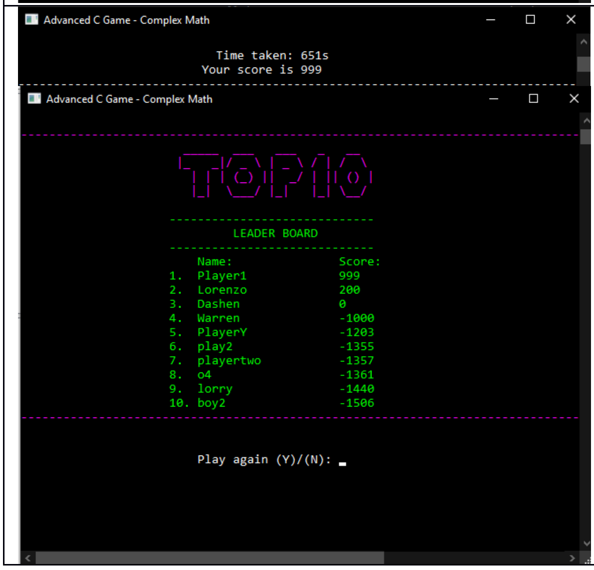


```
                     Please enter your name: Player1
                      Welcome to the game Player1 !

---------------------------------------------------------------------------------
--

                          BEGIN GAME? (Y): Y

                            Question 1/30 :
                      (-4+2i)x(2-9i) = 10+40i
                               CORRECT!

                            Question 2/30 :
                      (-10+8i)x(1-5i) = 30+58i
                               CORRECT!

                            Question 3/30 :
                      (9-8i)-(5+9i) = 1
                                WRONG!
                          Correct Answer = 4-17i

                            Question 4/30 :
                      (-10+9i)+(5-5i) = 2
                                WRONG!
                          Correct Answer = -5+4i

                            Question 5/30 :
                      (9-5i)-(9-10i) = 
```

Single player questions:

A player will have to answer 30 questions. If they get a question wrong an alert will be displayed along with the correct answer.

If they get the question correct, then they are notified as well.

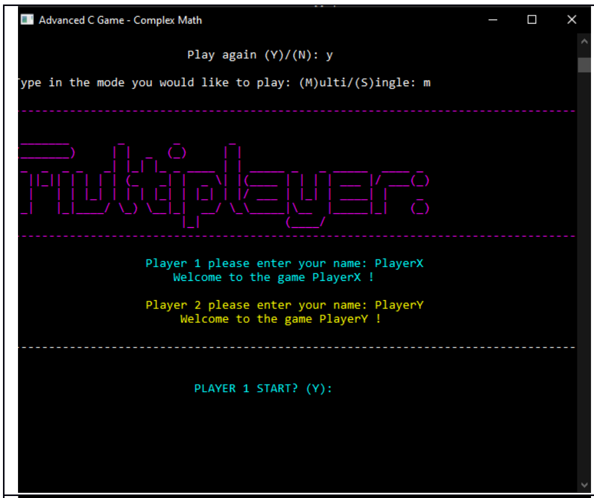

Advanced C Game - Complex Math
Advanced C Game - Complex Math
TOP10

```
                         Time taken: 651s
                       Your score is 999
---------------------------------------------------------------------------------
```

```
---------------------------------------------------------------------------------

                    -----------------------------
                            LEADER BOARD
                    -----------------------------
                        Name:                Score:
                    1.  Player1              999
                    2.  Lorenzo              200
                    3.  Dashen               0
                    4.  Warren               -1000
                    5.  PlayerY              -1203
                    6.  play2                -1355
                    7.  playertwo            -1357
                    8.  o4                   -1361
                    9.  lorry                -1440
                    10. boy2                 -1506
---------------------------------------------------------------------------------


                        Play again (Y)/(N): 
```

Single player total score and leader board:

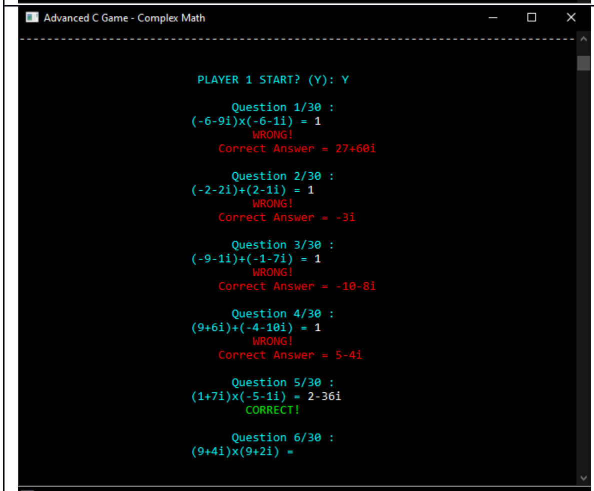
Once a player finishes all 30 questions, their total score, total time taken and their position on the leader board will be displayed.



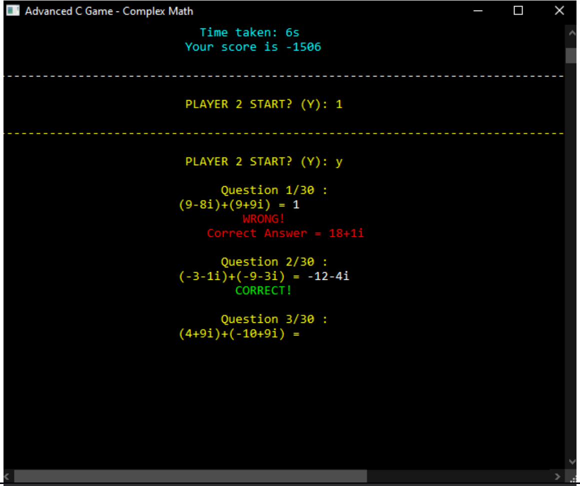
```
                    Play again (Y)/(N): y

ype in the mode you would like to play: (M)ulti/(S)ingle: m

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------

              Player 1 please enter your name: PlayerX
                  Welcome to the game PlayerX !

              Player 2 please enter your name: PlayerY
                   Welcome to the game PlayerY !

--------------------------------------------------------------------------------


                     PLAYER 1 START? (Y):
```

Multiplayer mode:

If the multiplayer mode is selected, then both players will need to fill in their respective names.



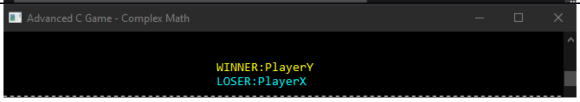
```
--------------------------------------------------------------------------------


                     PLAYER 1 START? (Y): Y

                          Question 1/30 :
                    (-6-9i)x(-6-1i) = 1
                             WRONG!
                        Correct Answer = 27+60i

                          Question 2/30 :
                    (-2-2i)+(2-1i) = 1
                             WRONG!
                        Correct Answer = -3i

                          Question 3/30 :
                    (-9-1i)+(-1-7i) = 1
                             WRONG!
                        Correct Answer = -10-8i

                          Question 4/30 :
                    (9+6i)+(-4-10i) = 1
                             WRONG!
                        Correct Answer = 5-4i

                          Question 5/30 :
                    (1+7i)x(-5-1i) = 2-36i
                            CORRECT!

                          Question 6/30 :
                    (9+4i)x(9+2i) =
```



```
                       Time taken: 6s
                     Your score is -1506

--------------------------------------------------------------------------------

                     PLAYER 2 START? (Y): 1

--------------------------------------------------------------------------------

                     PLAYER 2 START? (Y): y

                          Question 1/30 :
                     (9-8i)+(9+9i) = 1
                              WRONG!
                         Correct Answer = 18+1i

                          Question 2/30 :
                     (-3-1i)+(-9-3i) = -12-4i
                             CORRECT!

                          Question 3/30 :
                     (4+9i)+(-10+9i) =
```

Multiplayer player questions:

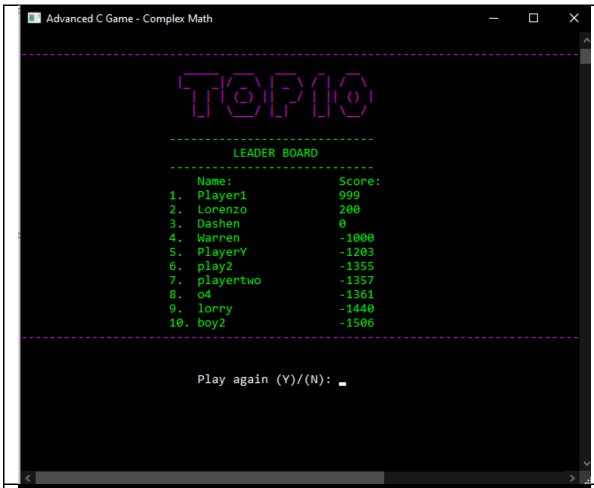
Both players will have their turn to answer their 30 questions. Player 1 will answer all the blue questions while player 2 will answer all the yellow questions.



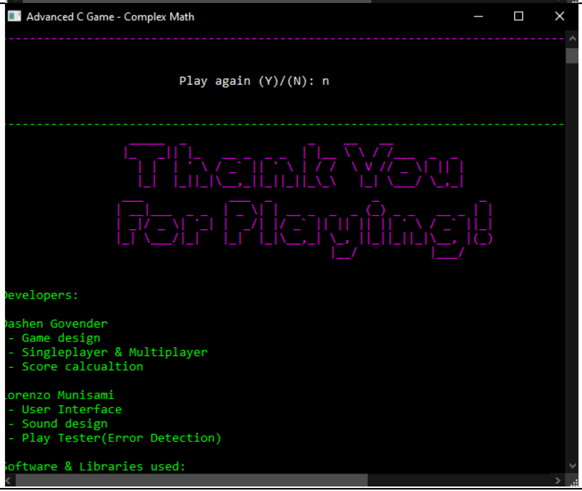
```
                          WINNER:PlayerY
                          LOSER:PlayerX
--------------------------------------------------------------------------------
```

Multiplayer results:

The winner and loser will be displayed as well as the player’s positions on the leader board if they make the top 10.



```
--------------------------------------------------------------------------------

                     -----------------------------
                              LEADER BOARD
                     -----------------------------
                         Name:                   Score:
                     1.  Player1                 999
                     2.  Lorenzo                 200
                     3.  Dashen                  0
                     4.  Warren                  -1000
                     5.  PlayerY                 -1203
                     6.  play2                   -1355
                     7.  playertwo               -1357
                     8.  o4                      -1361
                     9.  lorry                   -1440
                     10. boy2                    -1506
--------------------------------------------------------------------------------


                         Play again (Y)/(N): _
```



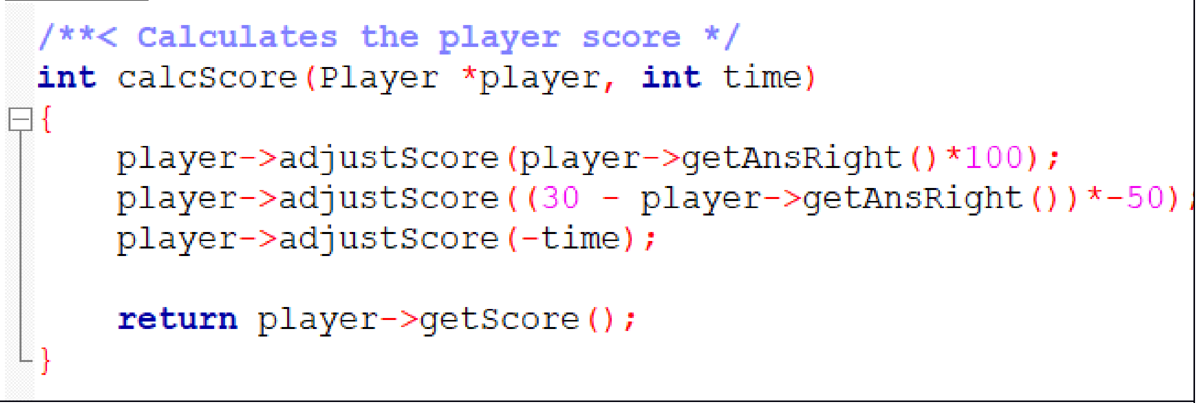
```
--------------------------------------------------------------------------------

                         Play again (Y)/(N): n

--------------------------------------------------------------------------------

Developers:

Dashen Govender
 - Game design
 - Singleplayer & Multiplayer
 - Score calcualtion

Lorenzo Munisami
 - User Interface
 - Sound design
 - Play Tester(Error Detection)

Software & Libraries used:
```

Credits Page:
If the player/s decide against playing again, then credits will be displayed.

Programming Techniques

1. Function

Screenshot:



```
/**< Calculates the player score */
int calcScore(Player *player, int time)
{
    player->adjustScore(player->getAnsRight()*100);
    player->adjustScore((30 - player->getAnsRight())*-50)
    player->adjustScore(-time);

    return player->getScore();
}
```

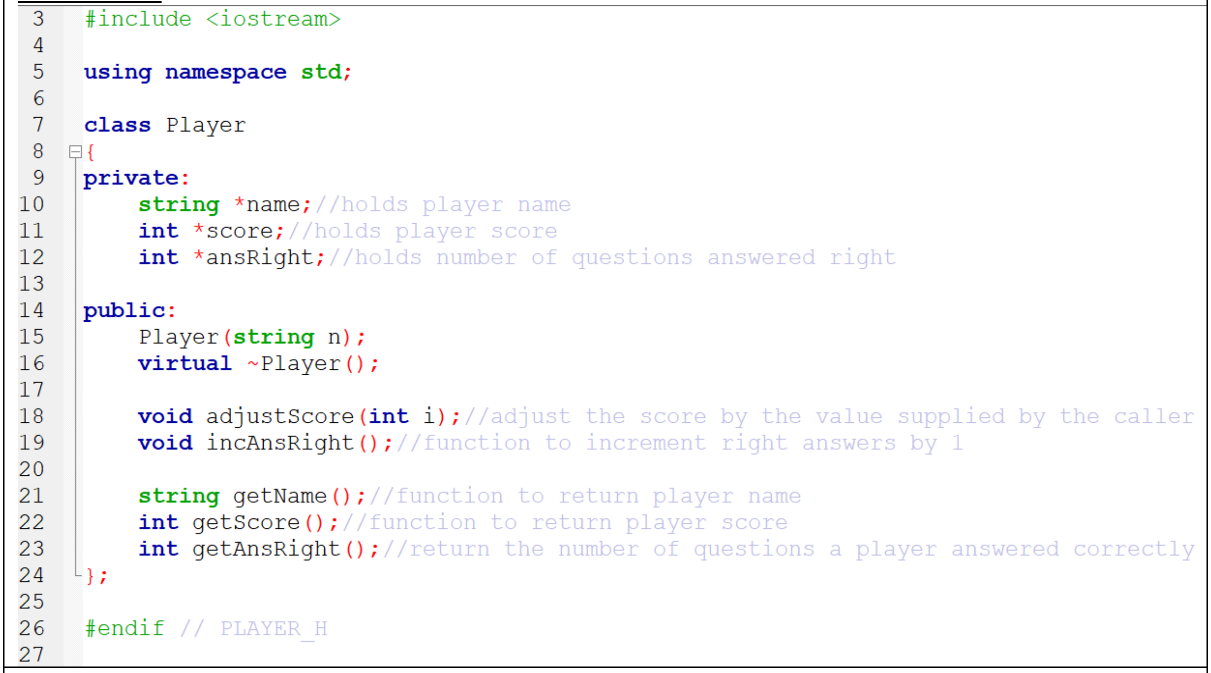
Motivation:

We created this function to calculate a player's score at the end of a turn. The player score is a fundamental part of Complex Math and it used repeatedly within the game and in many different scenarios. After analysis we found that the best solution was to create a function which would allow us to avoid repetition, when hardcoding, and to ensure easy maintenance, reusability and readability of the player scoring code. The function was made public to ensure reusability within the project scope.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	We have ensured that our functions: • provide encapsulation when needed • reduce the complexity of code to improve its reusability, maintenance and efficiency

2. Class

Screenshot:



```
#include <iostream>

using namespace std;

class Player
{
private:
    string *name;//holds player name
    int *score;//holds player score
    int *ansRight;//holds number of questions answered right

public:
    Player(string n);
    virtual ~Player();

    void adjustScore(int i);//adjust the score by the value supplied by the caller
    void incAnsRight();//function to increment right answers by 1

    string getName();//function to return player name
    int getScore();//function to return player score
    int getAnsRight();//return the number of questions a player answered correctly
};

#endif // PLAYER_H
```

Motivation:

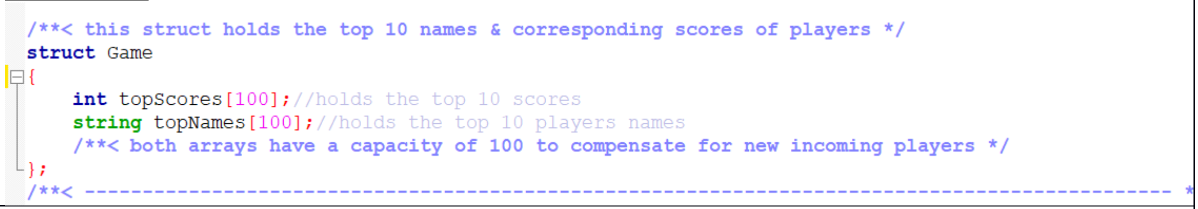
A player is an important object of our game as they have many associated data and behaviours. We decided to create a "Player" class to closely associate (group) the attributes and functions of a player together thus, allowing for easy implementation of encapsulation, maintainability, reusability, and scalability within our code. It also improves the readability of the code as all the player associated code will be found in the player class thus allowing specific player code to be found easily.

[*Explain why you decided to specifically use a class in this area of your code, and the functionality it provides to your code*

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	We created classes that ensure the concepts of encapsulation, maintainability, reusability, and scalability could be easily implemented.

3. Struct

Screenshot:



```
/**< this struct holds the top 10 names & corresponding scores of players */
struct Game
{
    int topScores[100];//holds the top 10 scores
    string topNames[100];//holds the top 10 players names
    /**< both arrays have a capacity of 100 to compensate for new incoming players */
};
/**< ---------------------------------------------------------------------------------------- *
```

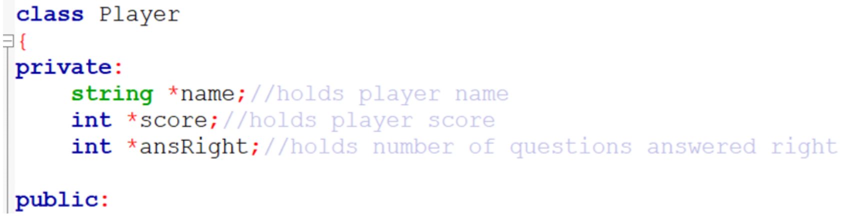
Motivation:

We stored our player's top scores in arrays which needed to be available to all the code in the main class. Our solution was to create a struct called "Game" which would ensure that the top score arrays would be accessible to all code that follows the struct definition. We did not have any associated functions for the arrays hence why there was no need for a class to be created.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	We have used the struct in a way that is efficient and effective as the top score arrays are needed in many areas of our code. It improves the reusability and the maintainability of our code.

4. Pointer

Screenshot:



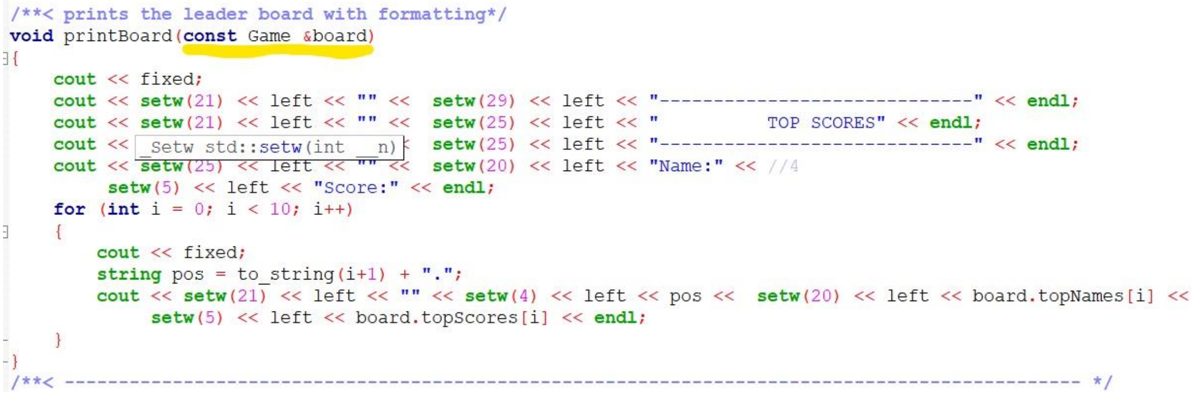
```
class Player
{
private:
    string *name;//holds player name
    int *score;//holds player score
    int *ansRight;//holds number of questions answered right

public:
```

Motivation:
A pointer has it's own memory address and size on the stack. The pointers above can be accessed quickly because the addresses for the pointers serve as indexes. We also do not need to make any copies of the fields when passing through functions. Pointers can have a null assignment, this is good for setting default values, such as 0 for scores. Having a pointer allows us to effectively delete fields and save memory space instead of reinitialising a field, where the previous value is lost and will take up space in memory.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	- Our pointers save time because Player is allocated on the heap, no copies are made, and fields are accessed accordingly. - The Player object can simply be removed and a new Player can be declared on the free store, not wasting memory space.

5. Reference

Screenshot:



```
/**< prints the leader board with formatting*/
void printBoard(const Game &board)
{
    cout << fixed;
    cout << setw(21) << left << "" <<  setw(29) << left << "------------------------------" << endl;
    cout << setw(21) << left << "" <<  setw(25) << left << "           TOP SCORES" << endl;
    cout << _Setw std::setw(int __n) <  setw(25) << left << "------------------------------" << endl;
    cout << setw(25) << left << "" <<  setw(20) << left << "Name:" << //4
         setw(5) << left << "Score:" << endl;
    for (int i = 0; i < 10; i++)
    {
        cout << fixed;
        string pos = to_string(i+1) + ".";
        cout << setw(21) << left << "" << setw(4) << left << pos <<  setw(20) << left << board.topNames[i] <<
             setw(5) << left << board.topScores[i] << endl;
    }
}
/**< ----------------------------------------------------------------------------------------------- */
```

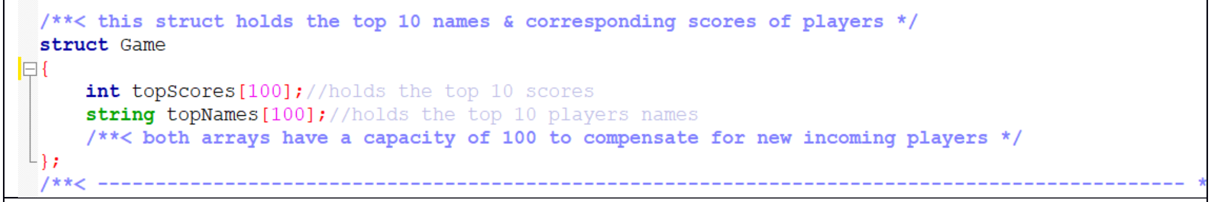
Motivation:

Using a reference in this example is better than using a pointer because all we have to do is show the reference symbol (&) instead of using dereferencing operators(*). This saves extra coding steps that would make the method look messy. We also want to pass by constant reference, because we are not changing the object, but it is large. Pass by value of large objects is costly. Pass by reference avoids copying and runs in minimal time since the address of the reference is the same as the variable it is referencing.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	-A reference to the struct Game is made constant, which is necessary in this method, since we need the structs members for printing only. -The code runs in minimal time, and referencing on the same address as Game uses less memory space as opposed to pointers. -Complexity of the code is minimal; we only add & and no dereferencing operators (*) are used, this is easier for the programmer to understand.

6. Data Structure

Screenshot:



```
/**< this struct holds the top 10 names & corresponding scores of players */
struct Game
{
    int topScores[100];//holds the top 10 scores
    string topNames[100];//holds the top 10 players names
    /**< both arrays have a capacity of 100 to compensate for new incoming players */
};
/**< ----------------------------------------------------------------------------------------------- *
```

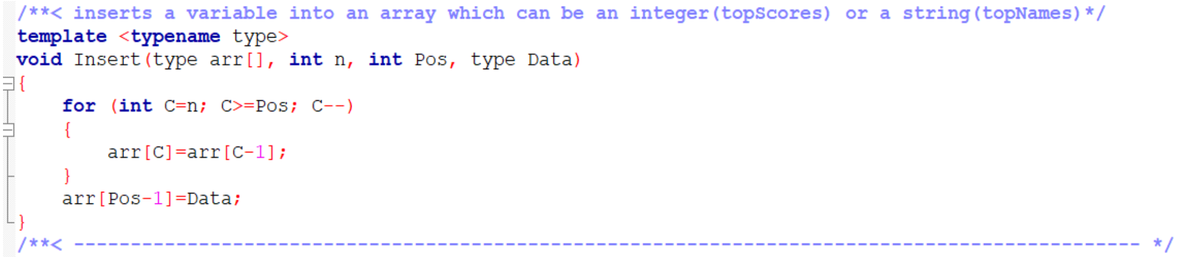
Motivation:

We needed to store the top players and their respective scores to implement a top 10 leader board. Our solution was to store the top players in a string array (topNames[]) and their respective top scores in an integer array (topScores[]). We decided to use arrays as we only need a fixed number of players within the top 10 leader board and arrays can easily sort in descending order. Accessing specific elements within an array is also very efficient thus replacing players in the array is very efficient as well. The arrays serves as a container for which their associated text files can be updated and stored.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	We used 2 different types of arrays, to keep track of our top players and their respective scores. This allows the arrays to be: • manipulated • printed • stored to text files • sorted

7. Function Template

Screenshot:



```
/**< inserts a variable into an array which can be an integer(topScores) or a string(topNames)*/
template <typename type>
void Insert(type arr[], int n, int Pos, type Data)
{
    for (int C=n; C>=Pos; C--)
    {
        arr[C]=arr[C-1];
    }
    arr[Pos-1]=Data;
}
/**< ------------------------------------------------------------------------------------------- */
```

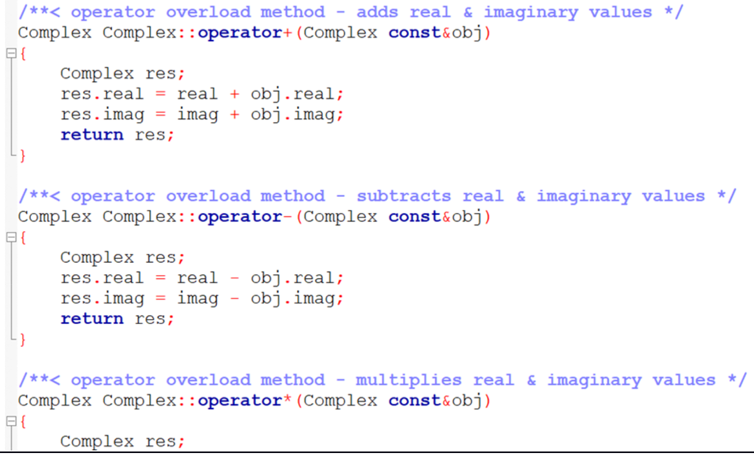
Motivation:

We needed to insert a player and their associated score into the topNames[] and the topScores[] arrays, if a player had gotten a score that was high enough to be on the top 10 leader board.
Our solution was to create one template function which would do the insertion for both the string topNames[]) and integer(topScores[]) arrays instead of creating 2 separate insert functions for each array, thus the function was made generic. This solution reduces repetitiveness as the sorting algorithm, used within the function, is the same for both arrays.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	Our template function effectively allowed us to solve 2 problems with one solution hence making it an effective and efficient solution to our insertion problem. It reduced the number of functions we had to create.

8. Operator Overloading

Screenshot:



```
/**< operator overload method - adds real & imaginary values */
Complex Complex::operator+(Complex const&obj)
{
    Complex res;
    res.real = real + obj.real;
    res.imag = imag + obj.imag;
    return res;
}

/**< operator overload method - subtracts real & imaginary values */
Complex Complex::operator-(Complex const&obj)
{
    Complex res;
    res.real = real - obj.real;
    res.imag = imag - obj.imag;
    return res;
}

/**< operator overload method - multiplies real & imaginary values */
Complex Complex::operator*(Complex const&obj)
{
    Complex res;
```

Motivation:

We needed to implement different functions for our complex math questions to evaluate and calculate solutions depending on the complex mathematical operator (+, -, *) used within the expression.

Our solution was to overload the operators, which provided a method for evaluating real and imaginary expressions automatically, which could not be done with ordinary mathematical operations.

How have you met the objectives?	Cross (X) the appropriate box	If you think that you have met the objective completely, provide a short explanation to support the claim
Not met		
Partially		
Completely	X	• We used operator overloading to make calculations for our complex number expressions, when regular mathematical operators would not have been useful. • Complex math questions serve as the basis of all our maths questions, it is therefore appropriate that it is implemented in this manner

Score Calculation

Complex Math records a player score. If the score is within the top 10 highest scores, then it is placed on our top 10 leader board. The calculation of the score is dependent on the number of questions answered right and wrong as well as the amount of time taken to answer all the questions.

Score Calculation:

For every question answered right a player earns 100 points and for every question answered incorrectly a player loses 50 points. In addition, a player will lose 1 point for every second they take to complete all 30 questions in the game (1s = -1 point).

e.g.) Player 1 gets 20 correct answers, 10 incorrect answers and takes a total of 102 seconds to answer all the questions.

Score = (20*100) – (10*50) – (102) = 1398

Leader Board:

An algorithm will then check if their score is high enough to be on the top 10 leader board and if so, then ‘player1’ will replace the position of a player, with a lower score, that results in the leader board being ordered numerically in descending order.

Multiplayer:

If there are 2 players facing off in the multiplayer mode, then their scores are compared resulting in a winner and loser, where the player with the highest score wins. If both players achieve the same score, then the result is a draw. Both players can still take their place on the leader board if they qualify for a place in the top 10.

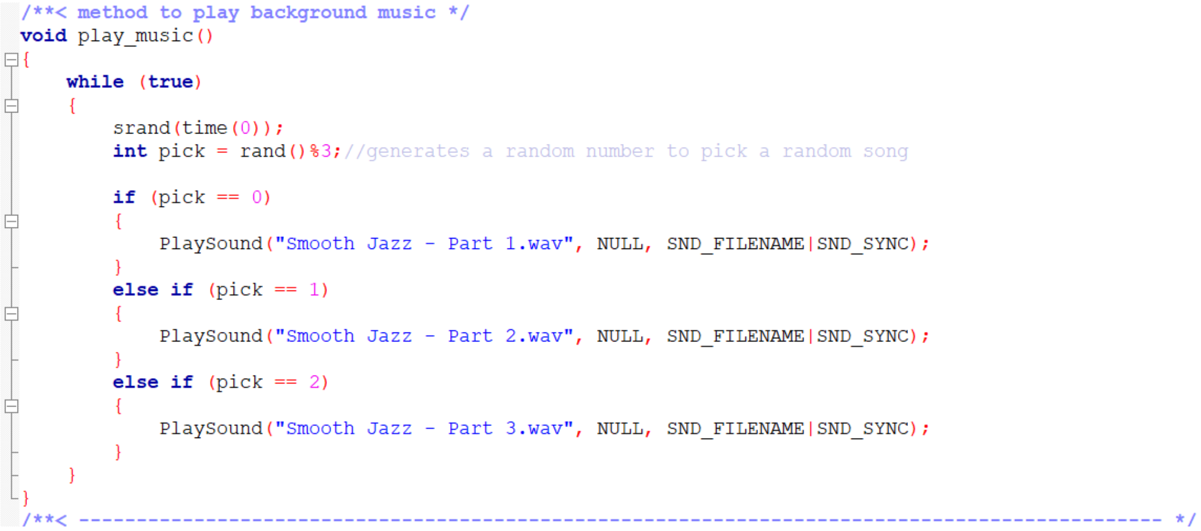
Additional Item

What does your quiz include?	Cross (X) the appropriate box
Sound	**X**
Threading	**X**
Multiplayer	**X**
Timer	**X**
Help and Credits	**X**
UI customisation	**X**

Additional Item 1 –

Sound

Complex Math has random music playing in the background. The genre of music is jazz which helps players relax and keep their focus when playing the game. One of the 3 songs will play from the start of the application up until the application is closed. This is made possible due to threading and the CTime library.

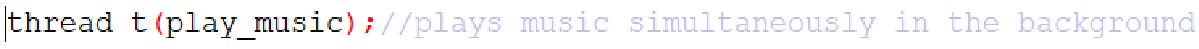


```
/**< method to play background music */
void play_music()
{
    while (true)
    {
        srand(time(0));
        int pick = rand()%3;//generates a random number to pick a random song

        if (pick == 0)
        {
            PlaySound("Smooth Jazz - Part 1.wav", NULL, SND_FILENAME|SND_SYNC);
        }
        else if (pick == 1)
        {
            PlaySound("Smooth Jazz - Part 2.wav", NULL, SND_FILENAME|SND_SYNC);
        }
        else if (pick == 2)
        {
            PlaySound("Smooth Jazz - Part 3.wav", NULL, SND_FILENAME|SND_SYNC);
        }
    }
}
/**< ----------------------------------------------------------------------------------------------- */
```

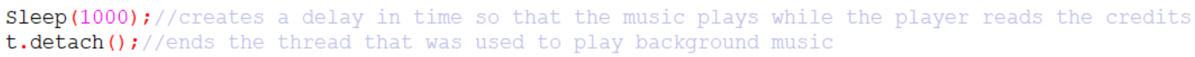
Additional Item 2 –

Threading

Complex Math uses threading which enables us to run multiple processes concurrently on segments of code. Threading makes it possible for background music and the game logic to execute simultaneously.



```
thread t(play_music);//plays music simultaneously in the background
```

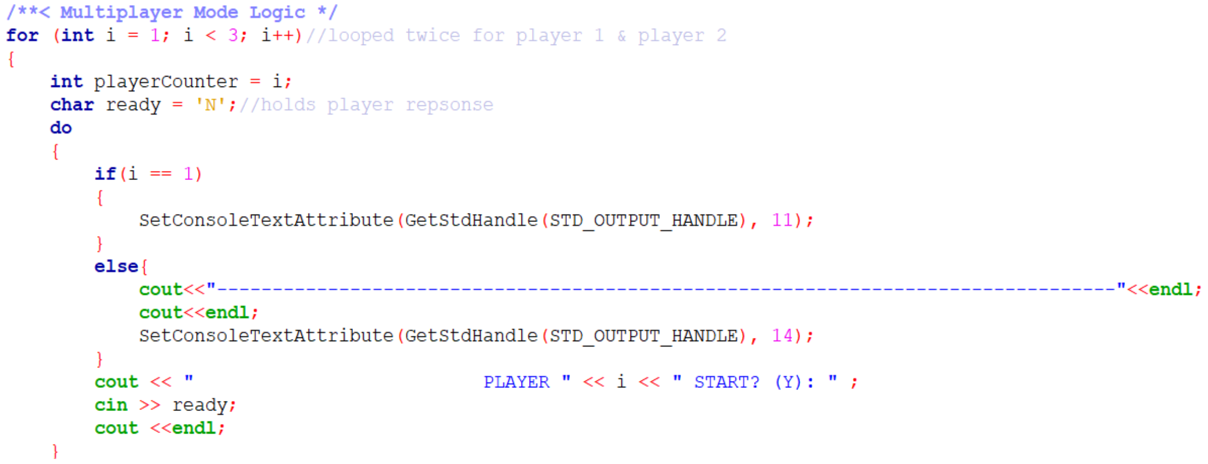


```
Sleep(1000);//creates a delay in time so that the music plays while the player reads the credits
t.detach();//ends the thread that was used to play background music
```

Additional Item 3 –

Multiplayer

Complex Math features a multiplayer mode which pits 2 local players against each other. Each player will have a turn to answer their 30 questions and the player with the highest score wins the duel. This competitive and fun mode promotes more interactive ways of learning mathematics and helps keep the learner's attention.

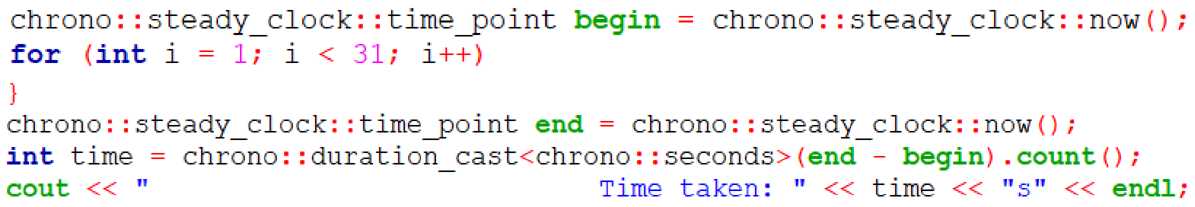


```
/**< Multiplayer Mode Logic */
for (int i = 1; i < 3; i++)//looped twice for player 1 & player 2
{
    int playerCounter = i;
    char ready = 'N';//holds player repsonse
    do
    {
        if(i == 1)
        {
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);
        }
        else{
            cout<<"--------------------------------------------------------------------------------"<<endl;
            cout<<endl;
            SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 14);
        }
        cout << "                            PLAYER " << i << " START? (Y): " ;
        cin >> ready;
        cout <<endl;
    }
```

Additional Item 4 –

Timer

Complex Math includes a timer which tracks the amount of time a player takes to answer all 30 questions of the quiz. This feature adds another layer of skill to the game as it tests the player's ability to answer questions as quickly and as correctly as possible. Time is taken into consideration when calculating a player's score thus challenging users to complete the quiz in an as little time as possible. The Chrono and CTime library were used to calculate time in Complex Math.

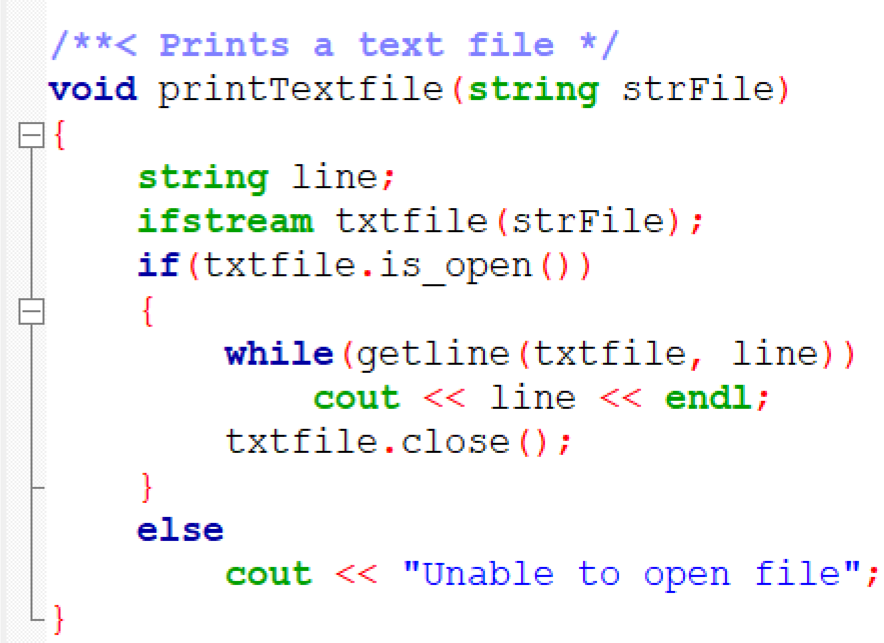


```
chrono::steady_clock::time_point begin = chrono::steady_clock::now();
for (int i = 1; i < 31; i++)
}
chrono::steady_clock::time_point end = chrono::steady_clock::now();
int time = chrono::duration_cast<chrono::seconds>(end - begin).count();
cout << "                         Time taken: " << time << "s" << endl;
```

Additional Item 5 –

Help and Credits

Complex Math features a Help and Credits page. This is implemented using a text file. The Help page explains the rules of the game, and the Credits page displays the creators of the game and the tools they used to create it.

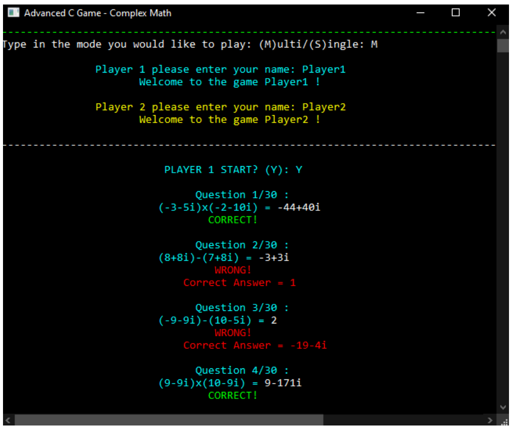


```cpp
/**< Prints a text file */
void printTextfile(string strFile)
{
    string line;
    ifstream txtfile(strFile);
    if(txtfile.is_open())
    {
        while(getline(txtfile, line))
            cout << line << endl;
        txtfile.close();
    }
    else
        cout << "Unable to open file";
}
```

Additional Item 6 –

User Interface Customisation

Complex Math makes use of arcade styled headings, bright colours, and larger fonts to present an easy to read, engaging and attractive user interface unlike the standard console format. Different sections of the game are displayed in different colours which makes it easier for the player to navigate through the game. In the multiplayer mode player 1 (aqua) and player 2 (yellow) are coloured differently which enables the players to easily discern player1-questions from player2-questions. The <windows.h> library was used to make this possible.



```
-----------------------------------------------------------------------------
Type in the mode you would like to play: (M)ulti/(S)ingle: M

              Player 1 please enter your name: Player1
                     Welcome to the game Player1 !

              Player 2 please enter your name: Player2
                     Welcome to the game Player2 !

-----------------------------------------------------------------------------

                         PLAYER 1 START? (Y): Y

                              Question 1/30 :
                        (-3-5i)x(-2-10i) = -44+40i
                                 CORRECT!

                              Question 2/30 :
                        (8+8i)-(7+8i) = -3+3i
                                  WRONG!
                            Correct Answer = 1

                              Question 3/30 :
                        (-9-9i)-(10-5i) = 2
                                  WRONG!
                            Correct Answer = -19-4i

                              Question 4/30 :
                        (9-9i)x(10-9i) = 9-171i
                                 CORRECT!
```

```
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 11);//sets text to blue
cout << "                                   Your score is " << calcScore(player1, time) << endl;
cout<<endl;
SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 15);//sets text to white
```

Appendix