Unit 2 – Darts Assignment

Please complete all the pages on this form and upload it along with your .exe, code and report. This form **must not be zipped.**

Please ✓ which tasks you have completed:

Part 1	✓
Part 2	✓
Part 3 A	✓
Part 3 B	✓

Files for uploading – these must be checked off before uploading

	✓ **to indicate these have been included**
Code – **must be compressed as a .zip**	✓
.exe – **this must be included in the .zip**	✓
Report - **.pdf must be uploaded separately from the .zip**	✓

Please answer the following questions

Questions	
Does the code compile without syntax errors? If no explain what the problems are and how you've attempted to resolve it.	yes
How many matches does your program simulate?	user decides
Who plays first?	user decides
What percentage accuracy have you assigned to the players?	user decides

Part 1

Explain how your code simulates a 301 game and how the frequencies are calculated and displayed. If you have not managed to achieve this, explain in detail the steps you've taken and where the problems have occurred.

It starts off by asking the user how many games they want to simulate. After the user inputs a number it asks who they want to go first player 1 or player 2. It makes whoever goes first throw dart at bull 50 and then switches players. Players throw at bull 50 until they get to score that around less then or equal to 20 away from bull. Then they for the difference between their score and 50 using single throws. Once they reach 50, they throw for bull again to win. The chances of hitting inner bullseye and single throws are changeable by the user. If they miss a bullseye, they hit a random number from 1 to 20. If they miss a single throw, they hit either the left or right neighbour of the target. In the end in displays the hit chances for single throws and bullseye throws, the amount of wins each player had and what percentages of games they won. I calculated the hit chances for the throws by incrementing different variables after each successful and failed throw which I then divided by the number of total throws made and multiplied by 100. The amount of games won was calculated by dividing the games won by the games played and multiplying by 100.

Part 2 – if attempted/completed

Explain how your code calculates the frequency across all of your simulations. If no frequency is calculated explain how you have modified your code from the 301 exercise. Please include a screen shot of the output.

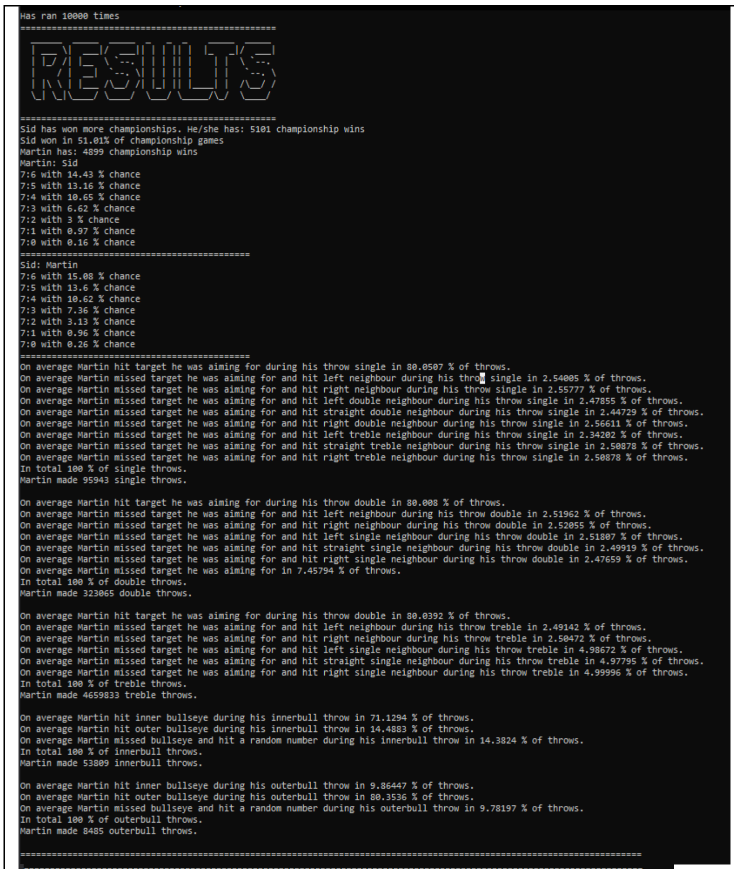
In 501 the user can also throw for double, treble, outer bull and has improved functions for throwing single and inner bull. Unlike in 301 when throwing an inner bull in 501 there is also a chance for the user to the outer bull. The chance of that happening 50% so if a player throws with 80% accuracy of hitting the inner bull, they will hit the outer bull in 10 % of their throws and a random number in another 10%. This is done by generating a random number between 1 and 2 which is used to decide what gets hit if the inner bull throw misses. The outer bull works the same way but instead hitting either inner bull or a random number. When making a single throw there is 12.5% chance of hitting one of the neighbours surrounding the target. So, if the user misses their single 20 throw, they have 12.5% chance of hitting either their left/right neighbour, left/centre/right double neighbour or left/centre/right treble neighbour. This is done by generating a random number between 1 and 8 which represent the 8 possible segments which might get hit if the shot misses. My double and treble function work the same way however in them the chances are different because the different segments surrounding the target. If the double throw misses it has 37.5% chance of missing the dartboard completely 25% chance to hit left/right double neighbour and a 37.5% chance to hit a left/centre/right single neighbour. If treble misses it has 75% chance to hit one of the top/bottom left/centre/right single neighbour and a 25% chance to hit a left/right treble neighbour. The frequencies of hits are calculated the same way as in 301, but now I also calculate the chances of the champions finishing with a certain amount of set wins per player by incrementing an array called setWonArray which hold 2 rows and 7 columns. The rows represent the players, while the columns represented the amount of set wins the enemy had. For example, if a game finishes 7:5 for player 1 I increment the 5th column in the 1st row. In the end of a game I use 2 for loops one which loops through the 2 rows and one which loops through the 6 columns in order to get the data out, divide it by the number of games then multiply it by 100 and save the resulting float in 2nd array called setsWonTotal which also has 2 rows and 7 columns and get displayed the to the console.

Part 3A – if attempted/completed

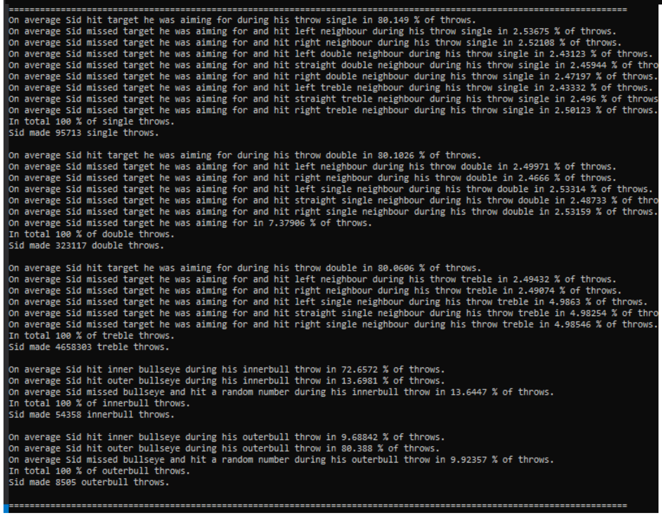
Explain how you have developed your solution beyond the basic algorithm in part 2. How does a player now decide on the best target to aim for? What challenges did this present? Please include a screen shot of the frequency output.

I used a checkout chart for 501 I found online. In it I saw that for all scores over 139 the best first dart to throw is treble 20. So, in my game the players throw for treble 20 until they get to under 140. Afterwards I use an array called threeDarts that holds 2 rows and 40 columns. The 2nd row holds scores from 139 to 101 and 99 while the 1st row holds the best possible target for that score. Using if else statements the score gets checked if it is within the range of the threeDarts array. If it is, I use a for loop which loops 40 times to go through the 2nd row until a value from there matches the current score of the player, then I get the data out of the first column and save it in a variable called target. I use target to make the throw and since in the checkout chart above 60 the first dart is always treble (with an exception at score 125 where the target is 25 outer bull) I make It throw for treble. After it completes the throw it runs strategy again on its next throw and compares the next value.
If the score = 125 instead of going through the threeDarts array it throws an outer bull.
Once the score gets between 107 and 61 it checks if the player is ahead. If he is not it checks if the current score – 50 is less than or equal to 60 and dividable by 3 or less than or equal to 40 and dividable by 2 so the algorithm knows whether it can throw a double or a treble to get the score to 50 and finish with a bull using only 2 darts.
If the score = 100 or is <= 98 and >= 61 it uses the twoDarts array which holds 2 rows and 39 columns and works, the same way as threeDarts but holds different targets. It uses a loop to compare the score to the value stored in the array and then throws for the target.
If the score is <= 60 or >= 51 it checks if the player is ahead again. If the player is ahead compared to the other one it continues playing safely and try to make the score go down to 40 by targeting the current score – 40 with a single throw so a win with a double is possible. If the player is not ahead it goes for a more aggressive strategy where it tries to get the score to 50 using the current score – 50 as a target so it can finish with a bull.
If the score = 50 it throws for a bull.
If the score is >= 41 and <= 49 it throws the dart targeting the current score-40 with a single throw to get the score in a range to finish with a double.
If the score is <= 40 it checks if it is even. If it is even it throws for double to win. If the score is odd it throws for 1 using a single throw to make it even.
All throws get run through a function which checks whether they are valid. The function checks and returns true if the score after the throw will be more than -1 and is not 1 after the throw is made. It also checks if the score of 0 is achieved using double or bull. If the function returns false, the score of the players get set to a temporary score which was equal to the score before the throw.
Main challenge I faced was figuring out how darts checkout strategies work, since I had never played darts before, and how to implement them in my program.

Screenshots of frequency output is on next page.



```
Has ran 10000 times
================================================
RESULTS
================================================
Sid has won more championships. He/she has: 5101 championship wins
Sid won in 51.01% of championship games
Martin has: 4899 championship wins
Martin: Sid
7:6 with 14.43 % chance
7:5 with 13.16 % chance
7:4 with 10.65 % chance
7:3 with 6.62 % chance
7:2 with 3 % chance
7:1 with 0.97 % chance
7:0 with 0.16 % chance
===========================================
Sid: Martin
7:6 with 15.08 % chance
7:5 with 13.6 % chance
7:4 with 10.62 % chance
7:3 with 7.36 % chance
7:2 with 3.13 % chance
7:1 with 0.96 % chance
7:0 with 0.26 % chance
===========================================
On average Martin hit target he was aiming for during his throw single in 80.0507 % of throws.
On average Martin missed target he was aiming for and hit left neighbour during his throw single in 2.54005 % of throws.
On average Martin missed target he was aiming for and hit right neighbour during his throw single in 2.55777 % of throws.
On average Martin missed target he was aiming for and hit left double neighbour during his throw single in 2.47855 % of throws.
On average Martin missed target he was aiming for and hit straight double neighbour during his throw single in 2.44729 % of throws.
On average Martin missed target he was aiming for and hit right double neighbour during his throw single in 2.56611 % of throws.
On average Martin missed target he was aiming for and hit left treble neighbour during his throw single in 2.34202 % of throws.
On average Martin missed target he was aiming for and hit straight treble neighbour during his throw single in 2.50878 % of throws.
On average Martin missed target he was aiming for and hit right treble neighbour during his throw single in 2.50878 % of throws.
In total 100 % of single throws.
Martin made 95943 single throws.

On average Martin hit target he was aiming for during his throw double in 80.008 % of throws.
On average Martin missed target he was aiming for and hit left neighbour during his throw double in 2.51962 % of throws.
On average Martin missed target he was aiming for and hit right neighbour during his throw double in 2.52055 % of throws.
On average Martin missed target he was aiming for and hit left single neighbour during his throw double in 2.51807 % of throws.
On average Martin missed target he was aiming for and hit straight single neighbour during his throw double in 2.49919 % of throws.
On average Martin missed target he was aiming for and hit right single neighbour during his throw double in 2.47659 % of throws.
On average Martin missed target he was aiming for in 7.45794 % of throws.
In total 100 % of double throws.
Martin made 323065 double throws.

On average Martin hit target he was aiming for during his throw double in 80.0392 % of throws.
On average Martin missed target he was aiming for and hit left neighbour during his throw treble in 2.49142 % of throws.
On average Martin missed target he was aiming for and hit right neighbour during his throw treble in 2.50472 % of throws.
On average Martin missed target he was aiming for and hit left single neighbour during his throw treble in 4.98672 % of throws.
On average Martin missed target he was aiming for and hit straight single neighbour during his throw treble in 4.97795 % of throws.
On average Martin missed target he was aiming for and hit right single neighbour during his throw treble in 4.99996 % of throws.
In total 100 % of treble throws.
Martin made 4659833 treble throws.

On average Martin hit inner bullseye during his innerbull throw in 71.1294 % of throws.
On average Martin hit outer bullseye during his innerbull throw in 14.4883 % of throws.
On average Martin missed bullseye and hit a random number during his innerbull throw in 14.3824 % of throws.
In total 100 % of innerbull throws.
Martin made 53809 innerbull throws.

On average Martin hit inner bullseye during his outerbull throw in 9.86447 % of throws.
On average Martin hit outer bullseye during his outerbull throw in 80.3536 % of throws.
On average Martin missed bullseye and hit a random number during his outerbull throw in 9.78197 % of throws.
In total 100 % of outerbull throws.
Martin made 8485 outerbull throws.

==========================================================================================================
```

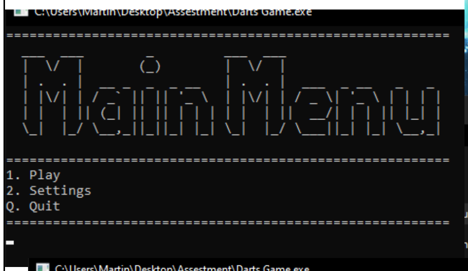


```
==========================================================================================================
On average Sid hit target he was aiming for during his throw single in 80.149 % of throws.
On average Sid missed target he was aiming for and hit left neighbour during his throw single in 2.53675 % of throws.
On average Sid missed target he was aiming for and hit right neighbour during his throw single in 2.52108 % of throws.
On average Sid missed target he was aiming for and hit left double neighbour during his throw single in 2.43123 % of throws.
On average Sid missed target he was aiming for and hit straight double neighbour during his throw single in 2.45944 % of thro
On average Sid missed target he was aiming for and hit right double neighbour during his throw single in 2.47197 % of throws.
On average Sid missed target he was aiming for and hit left treble neighbour during his throw single in 2.43332 % of throws.
On average Sid missed target he was aiming for and hit straight treble neighbour during his throw single in 2.496 % of throws
On average Sid missed target he was aiming for and hit right treble neighbour during his throw single in 2.50123 % of throws.
In total 100 % of single throws.
Sid made 95713 single throws.

On average Sid hit target he was aiming for during his throw double in 80.1026 % of throws.
On average Sid missed target he was aiming for and hit left neighbour during his throw double in 2.49971 % of throws.
On average Sid missed target he was aiming for and hit right neighbour during his throw double in 2.4666 % of throws.
On average Sid missed target he was aiming for and hit left single neighbour during his throw double in 2.53314 % of throws.
On average Sid missed target he was aiming for and hit straight single neighbour during his throw double in 2.48733 % of thro
On average Sid missed target he was aiming for and hit right single neighbour during his throw double in 2.53159 % of throws.
On average Sid missed target he was aiming for in 7.37906 % of throws.
In total 100 % of double throws.
Sid made 323117 double throws.

On average Sid hit target he was aiming for during his throw double in 80.0606 % of throws.
On average Sid missed target he was aiming for and hit left neighbour during his throw treble in 2.49432 % of throws.
On average Sid missed target he was aiming for and hit right neighbour during his throw treble in 2.49074 % of throws.
On average Sid missed target he was aiming for and hit left single neighbour during his throw treble in 4.9863 % of throws.
On average Sid missed target he was aiming for and hit straight single neighbour during his throw treble in 4.98254 % of thro
On average Sid missed target he was aiming for and hit right single neighbour during his throw treble in 4.98546 % of throws.
In total 100 % of treble throws.
Sid made 4658303 treble throws.

On average Sid hit inner bullseye during his innerbull throw in 72.6572 % of throws.
On average Sid hit outer bullseye during his innerbull throw in 13.6981 % of throws.
On average Sid missed bullseye and hit a random number during his innerbull throw in 13.6447 % of throws.
In total 100 % of innerbull throws.
Sid made 54358 innerbull throws.

On average Sid hit inner bullseye during his outerbull throw in 9.68842 % of throws.
On average Sid hit outer bullseye during his outerbull throw in 80.388 % of throws.
On average Sid missed bullseye and hit a random number during his outerbull throw in 9.92357 % of throws.
In total 100 % of outerbull throws.
Sid made 8505 outerbull throws.

==========================================================================================================
```

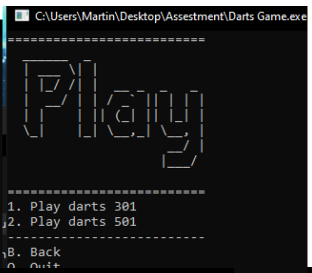
Part 3B – if attempted/completed

Explain how you developed the game beyond the basic interface in Part 2. How do users choose what to aim for? How did you design and develop the interface? Please include some screen shots of the user interaction.

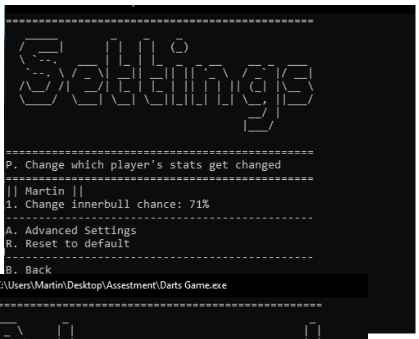
I have made it, so my program has menus. The first menu is the Main Menu from where you can either go to the settings menu or the play menu. In the settings menu the user can change the which player they are editing settings for, change the inner bull accuracy of the player, reset the settings back to what they entered when the program started, back to the Mein Menu or go into the advanced settings menu. In the advanced settings menu, they can change the outer bull, single, double and treble accuracy from their default 80 for each play, reset to the default values or back to the settings menu. In the play menu the user can go to the darts301 menu or the darts501 menu. In the darts301 menu they can start a simulation and chose whether they want the live information about each throw that is being made during the darts simulation to be displayed or not, which slow the simulation down. They can also choose whether they want extra frequency information being displayed after the simulation finishes. Darts501 menu is the same but it also has the option of starting an interactive game added. I designed the interface during the interactive game to show the current score of the player and the score of the ai in a easy to locate spot. It also shows the current score that is being made, whether it was successful, what it hit if it missed and the amount of games and sets won by the ai and the player.



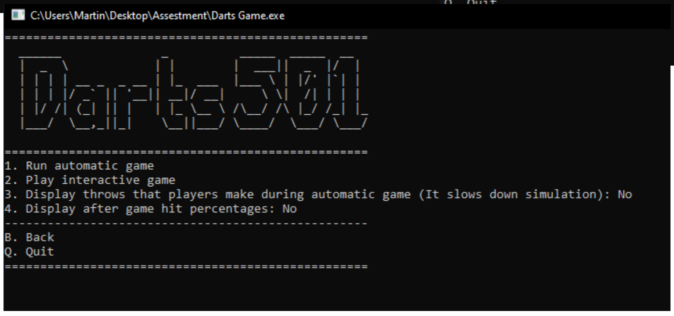
```
1. Play
2. Settings
Q. Quit
```



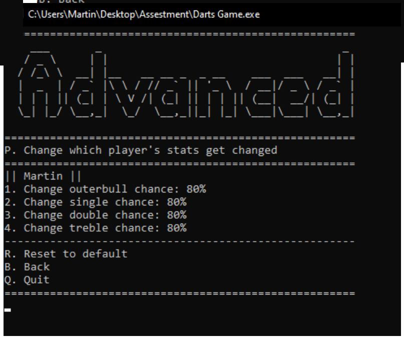
```
1. Play darts 301
2. Play darts 501
-------------------------
B. Back
Q. Quit
```



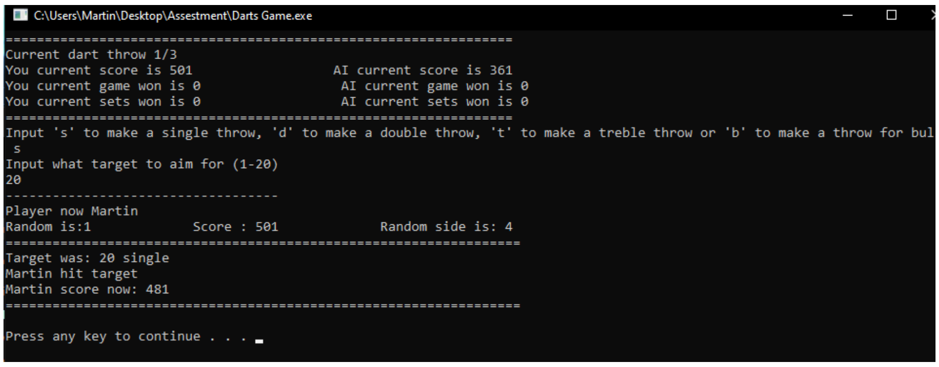
```
P. Change which player's stats get changed
|| Martin ||
1. Change innerbull chance: 71%
----------------------------------------
A. Advanced Settings
R. Reset to default
----------------------------------------
B. Back
```



```
1. Run automatic game
2. Play interactive game
3. Display throws that players make during automatic game (It slows down simulation): No
4. Display after game hit percentages: No
--------------------------------------------------
B. Back
Q. Quit
```



```
P. Change which player's stats get changed
|| Martin ||
1. Change outerbull chance: 80%
2. Change single chance: 80%
3. Change double chance: 80%
4. Change treble chance: 80%
----------------------------------------------
R. Reset to default
B. Back
Q. Quit
```



```
================================================================
Current dart throw 1/3
You current score is 501                  AI current score is 361
You current game won is 0                  AI current game won is 0
You current sets won is 0                  AI current sets won is 0
================================================================
Input 's' to make a single throw, 'd' to make a double throw, 't' to make a treble throw or 'b' to make a throw for bul
 s
Input what target to aim for (1-20)
20
-----------------------------------
Player now Martin
Random is:1               Score : 501              Random side is: 4
=================================================================
Target was: 20 single
Martin hit target
Martin score now: 481
=================================================================

Press any key to continue . . .
```

Code – for each class please copy this page and copy in the code. It does not matter if this goes over one page. Make sure the code is easily readable.

Class Name:
Players

.cpp

```
#include "Players.h"
#include <iostream>
#include <string>


//Constructors and Destructors

Players::Players(std::string n, int b) {
			name = n;
			innerbull_chance = b;
}
Players::Players()
{
			//Delete heap memory
			//Delete heap memory
			name = "no name";
}
Players::~Players()
{
			//delete heap memory
			//delete heap memory
			std::cout << "Destroyed object" << std::endl;
}

////Getters

//Stats getters
std::string Players::getName()
{
			return name;
}

//Chance Getters
int Players::getInnerbullChance()
{
			return innerbull_chance;
}
int Players::getOuterbullChance()
{
			return outerbull_chance;
}
int Players::getHitChance()
{
			return hit_chance;
}
int Players::getDoubleChance()
{
			return double_chance;
}
int Players::getTrebleChance()
{
			return treble_chance;
}
int Players::getWinCount()
{
			return winCount;
}
int Players::getSetsWon() {
			return setsWon;
}
int Players::getChampionshipsWon() {
			return championshipsWon;
}

//Score Getters
int Players::getScore()
{
			return score;
}
int Players::getTempScore()
{
			return tempScore;
}

//Info Getters
bool Players::getInfoThrows()
{
			return infoThrows;
}
bool Players::getInfoPercentages()
{
			return infoPercentages;
}
bool Players::getIsAhead()
{
		return isAhead;
}




//Setters
void Players::setName(std::string newName)
{
		name = newName;
}

//Chance setters
void Players::setInnerbullChance(int newInnerbullChance)
{
		innerbull_chance = newInnerbullChance;
}
void Players::setOuterbullChance(int newOuterbullChance)
{
		outerbull_chance = newOuterbullChance;
}
void Players::setHitChance(int newHitChance)
{
		hit_chance = newHitChance;
}
void Players::setDoubleChance(int newDoubleChance)
{
		double_chance = newDoubleChance;
}
void Players::setTrebleChance(int newTrebleChance)
{
		treble_chance = newTrebleChance;
}

//Stats setters
void Players::setWinCount(int newWinCount)
{
		winCount = newWinCount;
}
void Players::setSetsWon(int newSetsWon) {
		setsWon = newSetsWon;
}
void Players::setChampionshipsWon(int newChampionshipsWon) {
		championshipsWon = newChampionshipsWon;
}

//Score setters
void Players::setScore(int newScore)
{
		score = newScore;
}
void Players::setTempScore(int newTempScore)
{
		tempScore = newTempScore;
}

//Info setters
void Players::setInfoThrows(bool newInfoThrows)
{
		infoThrows = newInfoThrows;
}
void Players::setInfoPercentages(bool newInfoPercentages) {
		infoPercentages = newInfoPercentages;
}
void Players::setIsAhead(bool newAhead) {
		isAhead = newAhead;
}


//Throw functions
void Players::throwDart(int target, int gamemode, char trowType) { //Functions which allow the throwing of the dart using target, gamemode and trowType as variables
		tempScore = score; //Sets current score to temporary
		if (infoThrows == false) { //If user does not want to see throws
				std::cout.setstate(std::ios_base::failbit); //Stops couts from displaying
		}
		if (target == 50 && trowType == 'b') {
				throwInnerBull(gamemode);
		}
		else if (target == 25 && gamemode != 301 && trowType == 'b') {
				throwOuterBull(gamemode);
		}
		else if (target <= 20) {
				switch (trowType) {
				case 's':
						throwSingle(target, gamemode);
						break;
				case 'd':
						throwDouble(target, gamemode);
						break;
				case 't':
						throwTreble(target, gamemode);
						break;
				default:
						std::cout << "Invalid throw type. ERROR!" << std::endl;
```

```
                        break;
                    }
            }
            else {
                    std::cout << "Incorrect target. ERROR!" << target << std::endl;
            }
            if (infoThrows == false) { //If user does not want to see throws
                    std::cout.clear(); //Allows couts to display again
            }
}

void Players::throwInnerBull(int gamemode) {
            int random = rand() % 100 + 1; //Generate a random number between 1 and 100 and puts it in variable random
            int randomSide = rand() % 2 + 1; //Generate a random number between 1 and 2 and puts it in variable randomSide
            std::cout << "Player now " << name << std::endl;
            std::cout << "Random is:" << random;
            std::cout << "                  Score : " << score;
            std::cout << "                  Random side is: " << randomSide << std::endl;
            std::cout << "================================================================" << std::endl;
            std::cout << "Target was: 50 inner bull" << std::endl;

            if (random <= innerbull_chance) { //checks if generated number is smaller or equal to the innerbull chance. If it is hits. Otherwise if goes to next else if
                    std::cout << name << " hit 50 inner bullseye" << std::endl; //Displays message
                    if (validScore(50, gamemode, 'b')) { //Checks if the throw for 50 will be valid aka the score will not go below 0 if in darts301 and below 1 in darts501
                            score -= 50; //Reduces the score
                            std::cout << "Score now: " << score << std::endl;
                    }
                    else {
                            std::cout << "Invalid score" << std::endl;
                    }
                    innerBullseyesIN++; //Adds 1 to count
            }
            else if (randomSide == 1 && gamemode != 301) { //If the innerbullseye missed and the gamemode is 501 hit with 50% chance outerbullseye or a random number
from 1 to 20. Else if gamemode is 301 do not hit outerbullseye
                    std::cout << name << " hit 25 outer bullseye" << std::endl; //Displays message
                    if (validScore(25, gamemode, 'b')) { //Checks if the throw for 25 will be valid aka the score will not go below 0 if in darts301 and below 1 in darts501
                            score -= 25; //Reduces score
                            std::cout << "Score now: " << score << std::endl;
                    }
                    else {
                            std::cout << "Invalid score" << std::endl;
                    }
                    outerBullseyesIN++; //Adds 1 to count
            }
            else if (randomSide == 2 || gamemode == 301) { //50% chance to hit a random number from 1 to 20 if the gamemode is 501. 100% chance to hit a random number if
gamemode is 301
                    int random_score = rand() % 20 + 1; //Generates a random number between 1 and 20
                    std::cout << name << " hit " << random_score << ", missed inner bullseye" << std::endl; //Displays message
                    if (validScore(random_score, gamemode, 's')) { //Checks if the throw for random number will be valid aka the score will not go below 0 if in darts301
and below 1 in dart
                            score -= random_score;
                            std::cout << "Score now: " << score << std::endl;
                    }
                    else {
                            std::cout << "Invalid score" << std::endl;
                    }

                    missedBullseyesIN++; //Increments missed bullseyes
            }
            innerBullseyeThrows++; //Increments throws
            std::cout << "================================================================" << std::endl;
            std::cout << std::endl;
            //return false; old system
}

void Players::throwOuterBull(int gamemode) {
            if (gamemode != 301) { //Checks if gamemode is 301 if it is does not allow function to run
                    int random = rand() % 100 + 1; //Generate a random number between 1 and 100 and puts it in variable random
                    int randomSide = rand() % 2 + 1; //Generate a random number between 1 and 2 and puts it in variable randomSide
                    std::cout << "Player now " << name << std::endl;
                    std::cout << "Random is:" << random;
                    std::cout << "                  Score : " << score;
                    std::cout << "                  Random side is: " << randomSide << std::endl << std::endl;
                    std::cout << "================================================================" << std::endl;
                    std::cout << "Target was: 25 outer bull" << std::endl;

                    if (random <= outerbull_chance) { //checks if generated number is smaller or equal to the outerbull chance. If it is hits. Otherwise if goes to next else if
                            std::cout << name << " hit 25 outer bullseye" << std::endl; //Displays message
                            if (validScore(25, gamemode, 'b')) { //Checks if the throw for 25 will be valid aka the score will not go below 1 in darts 501
                                    score -= 25;
                                    std::cout << "Score now: " << score << std::endl;
                            }
                            else {
                                    std::cout << "Invalid score" << std::endl;
                            }
                            outerBullseyesOUT++;
                    }
                    else if (randomSide == 1) { //If the outerbullseye missed hit with 50% chance innerbulleye or a random number from 1 to 20.
                            std::cout << name << " hit 50 inner bullseye" << std::endl; //Displays message
                            if (validScore(50, gamemode, 'b')) { //Checks if the throw for 50 will be valid aka the score will not go below 1 in darts 501
                                    score -= 50;
                                    std::cout << "Score now: " << score << std::endl;
                            }
                            else {
                                    std::cout << "Invalid score" << std::endl;
            }
            innerBullseyesOUT++;
        }
        else  if (randomSide == 2) {
            int random_score = rand() % 20 + 1; //Generates a random number between 1 and 20
            std::cout << name << " hit " << random_score << ", missed outer bullseye" << std::endl; //Displays message
            if (validScore(random_score, gamemode, 's')) {
                score -= random_score;
                std::cout << "Score now: " << score << std::endl; //Checks if the throw for random number will be valid aka the score will
not go below 1 in darts 501
            }
            else {
                std::cout << "Invalid score" << std::endl;
            }

            missedBullseyesOUT++;
            //return false; old system
        }
        outerBullseyeThrows++; //Increments throws
        std::cout << "================================================================" << std::endl;
        std::cout << std::endl;
        /*std::cout.clear();*/
    }
    else {
        std::cout << "You cannot throw for an outer bull in 301 darts. Nice try!" << std::endl;
    }

}

void Players::throwSingle(int target, int gamemode) {
    int randomStraight = rand() % 100 + 1; //Generates  a random between 1 and 100
    int randomSide = rand() % 8 + 1; //Generates  a random between 1 and 8. Each number respresent one of the 8 surroundingg the target segments
    if (gamemode == 301) {
        randomSide = rand() % 2 + 1; //If gamemode is 301 generates  a random between 1 and 2. Because in 301 you cant hit double or treble
    }
    std::cout << "Player now " << name << std::endl;
    std::cout << "Random is:" << randomStraight;
    std::cout << "          Score : " << score;
    std::cout << "          Random side is: " << randomSide << std::endl;
    std::cout << "================================================================" << std::endl;
    if (randomStraight <= hit_chance) { //checks if the randomstraight number is less then or equal to hit_chance. If it is it hits. If it is not it misses hits a surrounding
target
        std::cout << "Target was: " << target << " single" << std::endl;
        if (validScore(target, gamemode, 's')) { //Checks if the target will be a valid score
            score -= target;
            std::cout << name << " hit target" << std::endl;
            std::cout << name << " score now: " << score << std::endl;
        }
        else {
            std::cout << name << " hit target" << std::endl;
            std::cout << "Invalid score" << std::endl;
        }
        hitTargetSIN++;
    }
    else {
        switch (randomSide) //Checks the randomSide number to know which one was hit.
        {
        case 1: //Hits left neighbour. 1/8 chance if gamemode is 501. 1/2 chance if gamemode is 301
            std::cout << "Target was: " << target << " single" << std::endl;
            std::cout << name << " missed single target, hit left single neighbour " << DartBoard::getLeftNeighbour(target) << std::endl;

            if (validScore(DartBoard::getLeftNeighbour(target), gamemode, 's')) { //Checks if score will be valid
                score -= DartBoard::getLeftNeighbour(target); //Reduces score using getLeftNeighbour function from dartboard
                std::cout << name << " score now: " << score << std::endl;
            }
            else {
                std::cout << "Invalid score" << std::endl;
            }
            wentLeftTargetSIN++;
            break;
        case 2: //Hits right neighbour. 1/8 chance if gamemode is 501. 1/2 chance if gamemode is 301
            std::cout << "Target was: " << target << " single" << std::endl;
            std::cout << name << " missed single target, hit right single neighbour " << DartBoard::getRightNeighbour(target) << std::endl;

            if (validScore(DartBoard::getRightNeighbour(target), gamemode, 's')) {
                score -= DartBoard::getRightNeighbour(target); //Reduces score using getRighrNeighbour function from dartboard
                std::cout << name << " score now: " << score << std::endl;
            }
            else {
                std::cout << "Invalid score" << std::endl;

            }
            wentRightTargetSIN++;
            break;
        case 3: //Hits top left neighbour. 1/8 chance
            std::cout << "Target was: " << target << " single" << std::endl;
            std::cout << "Missed single target, hit left double neighbour " << DartBoard::getLeftNeighbour(target) << std::endl;

            if (validScore(2 * DartBoard::getLeftNeighbour(target), gamemode, 'd')) {
                score -= 2 * DartBoard::getLeftNeighbour(target);
                std::cout << name << " score now: " << score << std::endl;
            }
            else {
                std::cout << "Invalid score" << std::endl;
            }
```

```
					wentLeftDoubleSIN++;
					break;
				case 4: //Hits neighbour on top. 1/8 chance
					std::cout << "Target was: " << target << " single" << std::endl;
					std::cout << "Missed single target, hit straight double neighbour " << target << std::endl;

					if (validScore(2 * target, gamemode, 'd')) {
						score -= 2 * target;
						std::cout << name << " score now: " << score << std::endl;
					}
					else {
						std::cout << "Invalid score" << std::endl;

					}
					wentStraightDoubleSIN++;
					break;
				case 5: //Hits neighbour on top right. 1/8 chance
					std::cout << "Target was: " << target << " single" << std::endl;
					std::cout << "Missed single target, hit right double neighbour " << DartBoard::getRightNeighbour(target) << std::endl;

					if (validScore(2 * DartBoard::getRightNeighbour(target), gamemode, 'd')) {
						score -= 2 * DartBoard::getRightNeighbour(target);
						std::cout << name << " score now: " << score << std::endl;
					}
					else {
						std::cout << "Invalid score" << std::endl;
					}
					wentRightDoubleSIN++;
					break;
				case 6: //Hits neighbour on bottom left. 1/8
					std::cout << "Target was: " << target << " single" << std::endl;
					std::cout << "Missed single target, hit left treble neighbour " << DartBoard::getLeftNeighbour(target) << std::endl;

					if (validScore(3 * DartBoard::getLeftNeighbour(target), gamemode, 't')) {
						score -= 3 * DartBoard::getLeftNeighbour(target);
						std::cout << name << " score now: " << score << std::endl;
					}
					else {
						std::cout << "Invalid score" << std::endl;

					}
					wentLeftTrebleSIN++;
					break;
				case 7: //Hits bottom neighbour. 1/8 chance
					std::cout << "Target was: " << target << " single" << std::endl;
					std::cout << "Missed single target, hit straight treble neighbour " << target << std::endl;

					if (validScore(3 * target, gamemode, 't')) {
						score -= 3 * target;
						std::cout << name << " score now: " << score << std::endl;
					}
					else {
						std::cout << "Invalid score" << std::endl;

					}
					wentStraightTrebleSIN++;
					break;
				case 8: //Hits bottom right neighbour
					std::cout << "Target was: " << target << " single" << std::endl;
					std::cout << "Missed single target, hit right treble neighbour " << DartBoard::getRightNeighbour(target) << std::endl;

					if (validScore(3 * DartBoard::getRightNeighbour(target), gamemode, 't')) {
						score -= 3 * DartBoard::getRightNeighbour(target);
						std::cout << name << " score now: " << score << std::endl;
					}
					else {
						std::cout << "Invalid score" << std::endl;
					}
					wentRightTrebleSIN++;
					break;
				default:
					std::cout << "ERROR IN SWITCH AT THROW SINGLE" << std::endl;
					break;
				}
		}
		throwsSIN++;
		std::cout << "=================================================================" << std::endl;
		std::cout << std::endl;
		/*std::cout.clear();*/
}

void Players::throwDouble(int target, int gamemode) {
		if (gamemode != 301) //Does not run unless gamemode is 501, because there is no double throws in 301
		{
				int randomStraight = rand() % 100 + 1; //Generate a random number between 1 and 100
				int randomSide = rand() % 8 + 1; //Generates  a random between 1 and 8. Each number respresent one of the 8 surroundingg the target segments
				std::cout << "Player now " << name << std::endl;
				std::cout << "Random is:" << randomStraight;
				std::cout << "				Score : " << score;
				std::cout << "				Random side is: " << randomSide << std::endl;
				std::cout << "=================================================================" << std::endl;
				std::cout << "Target was: " << target << " double" << std::endl;

				if (randomStraight <= double_chance) { //checks if generated number is smaller or equal to the outerbull chance. If it is hits. Otherwise if goes to next
else if
```

```
            std::cout << name << " hit target" << std::endl;
            if (validScore(2 * target, gamemode, 'd')) {
                score -= 2 * target;
                std::cout << name << " score now: " << score << std::endl;
            }
            else {
                std::cout << "Invalid score" << std::endl;
            }
            hitTargetDOU++;
        }
        else {
            switch (randomSide) //Checks the randomSide number to know which one was hit.
            {
            case 1:
                std::cout << name << " missed double target, hit left double neighbour " << DartBoard::getLeftNeighbour(target) <<
std::endl; //Hit left neighbour 1/8 chance

                if (validScore(2 * DartBoard::getLeftNeighbour(target), gamemode, 'd')) {
                    score -= 2 * DartBoard::getLeftNeighbour(target);
                    std::cout << name << " score now: " << score << std::endl;
                }
                else {
                    std::cout << "Invalid score" << std::endl;
                }
                wentLeftTargetDOU++;
                break;
            case 2:
                std::cout << name << " missed double target, hit right double neighbour " << DartBoard::getRightNeighbour(target) <<
std::endl; //Hit right neighbour 1/8 chance

                if (validScore(2 * DartBoard::getRightNeighbour(target), gamemode, 'd')) {
                    score -= 2 * DartBoard::getRightNeighbour(target);
                    std::cout << name << " score now: " << score << std::endl;
                }
                else {
                    std::cout << "Invalid score" << std::endl;

                }
                wentRightTargetDOU++;
                break;
            case 3:
            case 4: //Misses dartboard 3/8 chance because of possible 3 miss segments
            case 5:
                std::cout << "Missed dartboard" << std::endl;
                std::cout << name << " score now: " << score << std::endl;
                missed++;
                break;
            case 6: //Hits bottom left neighbour
                std::cout << name << " missed double target, hit left single neighbour " << DartBoard::getLeftNeighbour(target) <<
std::endl;

                if (validScore(DartBoard::getLeftNeighbour(target), gamemode, 's')) {
                    score -= DartBoard::getLeftNeighbour(target);
                    std::cout << name << " score now: " << score << std::endl;
                }
                else {
                    std::cout << "Invalid score" << std::endl;

                }
                wentLeftSingleDOU++;
                break;
            case 7: //Hits bottom neighbour
                std::cout << name << " missed double target, hit straight single neighbour " << target << std::endl;

                if (validScore(target, gamemode, 's')) {
                    score -= target;
                    std::cout << name << " score now: " << score << std::endl;
                }
                else {
                    std::cout << "Invalid score" << std::endl;

                }
                wentStraightSingleDOU++;
                break;
            case 8: //Hits bottom right neighbour
                std::cout << name << " missed double target, hit right single neighbour " << DartBoard::getRightNeighbour(target) <<
std::endl;

                if (validScore(DartBoard::getRightNeighbour(target), gamemode, 's')) {
                    score -= DartBoard::getRightNeighbour(target);
                    std::cout << name << " score now: " << score << std::endl;
                }
                else {
                    std::cout << "Invalid score" << std::endl;
                }
                wentRightSingleDOU++;
                break;
            default:
                std::cout << "ERROR IN SWITCH AT THROW DOUBLE" << std::endl;
                break;
            }
        }
        throwsDOU++;
        std::cout << "==============================================================" << std::endl;
        std::cout << std::endl;
```

```
		}
		else {
				std::cout << "You cannot throw for double in 301. Nice try!" << std::endl;
		}
}

void Players::throwTreble(int target, int gamemode) {
		if (gamemode != 301)  //Does not run unless gamemode is 501, because there is no treble throws in 301
		{
				int randomStraight = rand() % 100 + 1; //Generates  a random between 1 and 100.
				int randomSide = rand() % 8 + 1; //Generates  a random between 1 and 8. Each number respresent one of the 8 surroundingg the target segments
				std::cout << "Player now " << name << std::endl;
				std::cout << "Random is:" << randomStraight;
				std::cout << "				Score : " << score;
				std::cout << "				Random side is: " << randomSide << std::endl;
				std::cout << "=================================================================" << std::endl;
				std::cout << "Target was: " << target << " treble" << std::endl;

				if (randomStraight <= treble_chance) { //Checks if randomStraight is equal to less the treble chance. If it is the shot hits, otherwise it misses
						if (validScore(3 * target, gamemode, 't')) {
								score -= 3 * target;
								std::cout << name << " hit target" << std::endl;
								std::cout << name << " score now: " << score << std::endl;
						}
						else {
								std::cout << "Invalid score" << std::endl;
						}

						hitTargetTRE++;
				}
				else {
						switch (randomSide) //Checks the randomSide number to know which one was hit.
						{
						case 1: //Hit left neighbour 1/8 chance
								std::cout << "Missed treble target, hit left treble neighbour " << DartBoard::getLeftNeighbour(target) << std::endl;

								if (validScore(3 * DartBoard::getLeftNeighbour(target), gamemode, 't')) {
										score -= 3 * DartBoard::getLeftNeighbour(target);
										std::cout << name << " score now: " << score << std::endl;
								}
								else {
										std::cout << "Invalid score" << std::endl;
								}
								wentLeftTargetTRE++;
								break;
						case 2: //Hit right neighbour. 1/8 chance
								std::cout << "Missed treble target, hit right treble neighbour " << DartBoard::getRightNeighbour(target) << std::endl;

								if (validScore(3 * DartBoard::getRightNeighbour(target), gamemode, 't')) {
										score -= 3 * DartBoard::getRightNeighbour(target);
										std::cout << name << " score now: " << score << std::endl;
								}
								else {
										std::cout << "Invalid score" << std::endl;
								}
								wentRightTargetTRE++;
								break;
						case 3:
						case 4: //hit top/bot left single neighbour. 2/8 chance sinde treble segments are surrounded by single segments
								std::cout << "Missed treble target, hit left single neighbour " << DartBoard::getLeftNeighbour(target) << std::endl;

								if (validScore(DartBoard::getLeftNeighbour(target), gamemode, 's')) {
										score -= DartBoard::getLeftNeighbour(target);
										std::cout << name << " score now: " << score << std::endl;
								}
								else {
										std::cout << "Invalid score" << std::endl;

								}
								wentLeftSingleTRE++;
								break;
						case 5:
						case 6: //hit top/bot single neighbour. 2/8 chance sinde treble segments are surrounded by single segments
								std::cout << "Missed double target, hit straight single neighbour " << target << std::endl;

								if (validScore(target, gamemode, 's')) {
										score -= target;
										std::cout << name << " score now: " << score << std::endl;
								}
								else {
										std::cout << "Invalid score" << std::endl;

								}
								wentStraightSingleTRE++;
								break;
						case 7:
						case 8: //hit top/bot right single neighbour. 2/8 chance sinde treble segments are surrounded by single segments
								std::cout << "Missed double target, hit right single neighbour " << DartBoard::getRightNeighbour(target) << std::endl;

								if (validScore(DartBoard::getRightNeighbour(target), gamemode, 's')) {
										score -= DartBoard::getRightNeighbour(target);
										std::cout << name << " score now: " << score << std::endl;
								}
								else {
										std::cout << "Invalid score" << std::endl;
```

```
                }
                wentRightSingleTRE++;
                break;
            default:
                std::cout << "ERROR IN SWITCH AT THROW TREBLE" << std::endl;
                break;
            }
        }
        throwsTRE++;
        std::cout << "==================================================================" << std::endl;
        std::cout << std::endl;
    }
    else {
        std::cout << "You cannot throw for double in 301. Nice try!" << std::endl;
    }
}

//Strategy
void Players::strategy() {
    //std::cout << "STRATEGY" << std::endl;
    int difference = score - 50;

    if (score >= 140) { //if the score is over 140 throws for treble since treble is worth the most
        throwDart(20, 501, 't');
    }

    else if (isAhead == false && score <= 107 && score >= 61 && ((difference <= 60 && difference % 3 == 0) || (difference <= 40 && difference % 2 == 0))) { //Goes for a
risky throw if the player is not ahead, and score is between 107 and 61 where the score - 50(difference) is <= 40 and dividable by 2 or <= 60 and dividable bt 3. if it passes checks
throws a double or a treble which will allow it to get to 50 in one throw
        if (difference % 3 == 0) { //Checks if the score - 50 leaves a remainder. If it does not it runs
            int target = difference / 3;
            throwDart(target, 501, 't');
        }
        else if (difference % 2 == 0) { //Checks if the score - 50 leaves a remainder. If it does not it runs
            int target = difference / 2;
            throwDart(target, 501, 'd');
        }
        else {
            //std::cout << name << " Error in strategy. %checks" << std::endl;
        }
    }
    else if (score == 99 || (score <= 139 && score >= 101)) {
        for (int i = 0; i <= 39; i++) { //Loops until it finds a value from the array that is the same as the score in the threeDart array
            if (threeDart[1][i] == score) { //Runs if the value from the array is the same as the score
                int target = threeDart[0][i];
                if (target == 25) { //If the value for the target in the array is 25. Throw for outerbull, cause you throw a score of 25 using a
treble
                    throwDart(target, 501, 'b');
                    break;
                }
                else {
                    throwDart(target, 501, 't');
                    break;
                }
            }
        }
    }
    else if (score == 100 || (score <= 98 && score >= 61)) {
        for (int i = 0; i <= 38; i++) { //Loops until it finds a value from the array that is the same as the score in the threeDart array
            if (twoDart[1][i] == score) { //Runs if the value from the array is the same as the score
                int target = twoDart[0][i];
                throwDart(target, 501, 't');
                break;
            }
        }
    }
    else if (score <= 60 && score >= 51) { //If not ahead tries to make the score 50 so a throw for bull will win. If ahead goes for a normal double finish
        if (isAhead == true) {
            //std::cout << "Is ahead so go for double finish" << std::endl;
            throwDart(score - 40, 501, 's');
        }
        else {
            //std::cout << "Is not ahead so go for bull finish" << std::endl;
            throwDart(score - 50, 501, 's');
        }

    }
    else if (score == 50) {
        throwDart(50, 501, 'b');
    }
    else if (score >= 41 && score <= 49) { //Gets the odd number to even
        throwDart(score - 40, 501, 's');
    }
    else if (score % 2 == 0 && score <= 40) { //Trows double to win
        //std::cout << "SCORE IS EVEN!!!" << std::endl;
        for (int i = 0; i <= 19; i++) {
            if (oneDart[1][i] == score) {
                //std::cout << "Score is " << score << " which can be won with double" << std::endl;
                //std::cout << "Element found at index " << i << std::endl;
                //std::cout << oneDart[0][i] << std::endl;
                //std::cout << std::endl;
                int target = oneDart[0][i];
                throwDart(target, 501, 'd');
                break;
```

```
					}
				}
			}
			else {
				//std::cout << "SCORE IS NOT EVEN!!!" << std::endl;
				throwDart(1, 501, 's');
			}
}


//Validation functions
void Players::averageCalculate(int gamemode)
{
			////301 Stats
			if (gamemode == 301) { //Displays stats for 301 if gamemode is 301
				if (infoPercentages == true) {
					float averageHitTarget = (float(hitTargetSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " hit target he was aiming for during his throw single in " << averageHitTarget << " % of throws."
<< std::endl; //Displays message
					float averageLeftTarget = (float(wentLeftTargetSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit left neighbour during his throw single in " <<
averageLeftTarget << " % of throws." << std::endl; //Displays message
					float averageRightTarget = (float(wentRightTargetSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit right neighbour during his throw single in " <<
averageRightTarget << " % of throws." << std::endl; //Displays message

					float averageInnerBull = (float(innerBullseyesIN) / float(innerBullseyeThrows)) * 100;
					std::cout << "On average " << name << " hit inner bullseye during his innerbull throw in " << averageInnerBull << " % of throws." <<
std::endl; //Displays message
					float averageMissedBullseyes = (float(missedBullseyesIN) / float(innerBullseyeThrows)) * 100;
					std::cout << "On average " << name << " missed bullseye and hit a random number during his innerbull throw in " <<
averageMissedBullseyes << " % of throws." << std::endl; //Displays message

					float total = averageHitTarget + averageLeftTarget + averageRightTarget;
					std::cout << "In total " << total << " % of single throws." << std::endl; //Displays message
					std::cout << name << " made " << throwsSIN << " single throws." << std::endl;
					float totalBull = averageInnerBull + averageMissedBullseyes;
					std::cout << "In total " << totalBull << " % of innerbull throws." << std::endl; //Displays message
					std::cout << name << " made " << innerBullseyeThrows << " innerbull throws." << std::endl;
				}
				std::cout << name << " had " << winCount << " wins" << std::endl; //Displays message
				std::cout << std::endl;
			}
			else if (gamemode == 501) { //Displays stats for 501 if gamemode is 301
				if (infoPercentages == true) {
					//Single STATS
					float averageHitTargetSIN = (float(hitTargetSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " hit target he was aiming for during his throw single in " << averageHitTargetSIN << " % of
throws." << std::endl; //Displays message
					float averageLeftTargetSIN = (float(wentLeftTargetSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit left neighbour during his throw single in " <<
averageLeftTargetSIN << " % of throws." << std::endl; //Displays message
					float averageRightTargetSIN = (float(wentRightTargetSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit right neighbour during his throw single in " <<
averageRightTargetSIN << " % of throws." << std::endl; //Displays message

					float averageWentLeftDoubleSIN = (float(wentLeftDoubleSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit left double neighbour during his throw single in " <<
averageWentLeftDoubleSIN << " % of throws." << std::endl; //Displays message
					float averageWentStraightDoubleSIN = (float(wentStraightDoubleSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit straight double neighbour during his throw single in "
<< averageWentStraightDoubleSIN << " % of throws." << std::endl; //Displays message
					float averageWentRightDoubleSIN = (float(wentRightDoubleSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit right double neighbour during his throw single in " <<
averageWentRightDoubleSIN << " % of throws." << std::endl; //Displays message

					float averageWentLeftTrebleSIN = (float(wentLeftTrebleSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit left treble neighbour during his throw single in " <<
averageWentLeftTrebleSIN << " % of throws." << std::endl; //Displays message
					float averageWentStraightTrebleSIN = (float(wentStraightTrebleSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit straight treble neighbour during his throw single in " <<
averageWentStraightTrebleSIN << " % of throws." << std::endl; //Displays message
					float averageWentRightTrebleSIN = (float(wentRightTrebleSIN) / float(throwsSIN)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit right treble neighbour during his throw single in " <<
averageWentRightTrebleSIN << " % of throws." << std::endl; //Displays message

					float totalSingle = averageHitTargetSIN + averageLeftTargetSIN + averageRightTargetSIN + averageWentLeftDoubleSIN +
averageWentStraightDoubleSIN + averageWentRightDoubleSIN + averageWentLeftTrebleSIN + averageWentStraightTrebleSIN + averageWentRightTrebleSIN;
					std::cout << "In total " << totalSingle << " % of single throws." << std::endl; //Displays message
					std::cout << name << " made " << throwsSIN << " single throws."<< std::endl;
					std::cout << std::endl;

					//Double STATS
					float averageHitTargetDOU = (float(hitTargetDOU) / float(throwsDOU)) * 100;
					std::cout << "On average " << name << " hit target he was aiming for during his throw double in " << averageHitTargetDOU << " % of
throws." << std::endl; //Displays message
					float averageLeftTargetDOU = (float(wentLeftTargetDOU) / float(throwsDOU)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit left neighbour during his throw double in " <<
averageLeftTargetDOU << " % of throws." << std::endl; //Displays message
					float averageRightTargetDOU = (float(wentRightTargetDOU) / float(throwsDOU)) * 100;
					std::cout << "On average " << name << " missed target he was aiming for and hit right neighbour during his throw double in " <<
averageRightTargetDOU << " % of throws." << std::endl; //Displays message

					float averageWentLeftSingleDOU = (float(wentLeftSingleDOU) / float(throwsDOU)) * 100;
```

```
                        std::cout << "On average " << name << " missed target he was aiming for and hit left single neighbour during his throw double in " <<
averageWentLeftSingleDOU << " % of throws." << std::endl; //Displays message
                        float averageWentStraightSingleDOU = (float(wentStraightSingleDOU) / float(throwsDOU)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit straight single neighbour during his throw double in "
<< averageWentStraightSingleDOU << " % of throws." << std::endl; //Displays message
                        float averageWentRightSingleDOU = (float(wentRightSingleDOU) / float(throwsDOU)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit right single neighbour during his throw double in " <<
averageWentRightSingleDOU << " % of throws." << std::endl; //Displays message

                        float averageMissedDOU = (float(missed) / float(throwsDOU)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for in " << averageMissedDOU << " % of throws." << std::endl;
//Displays message

                        float totalDouble = averageHitTargetDOU + averageLeftTargetDOU + averageRightTargetDOU + averageWentLeftSingleDOU +
averageWentStraightSingleDOU + averageWentRightSingleDOU + averageMissedDOU;
                        std::cout << "In total " << totalDouble << " % of double throws." << std::endl; //Displays message
                        std::cout << name << " made " << throwsDOU << " double throws." << std::endl;
                        std::cout << std::endl;

                        //Treble STATS
                        float averageHitTargetTRE = (float(hitTargetTRE) / float(throwsTRE)) * 100;
                        std::cout << "On average " << name << " hit target he was aiming for during his throw double in " << averageHitTargetTRE << " % of
throws." << std::endl; //Displays message
                        float averageLeftTargetTRE = (float(wentLeftTargetTRE) / float(throwsTRE)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit left neighbour during his throw treble in " <<
averageLeftTargetTRE << " % of throws." << std::endl; //Displays message
                        float averageRightTargetTRE = (float(wentRightTargetTRE) / float(throwsTRE)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit right neighbour during his throw treble in " <<
averageRightTargetTRE << " % of throws." << std::endl; //Displays message

                        float averageWentLeftSingleTRE = (float(wentLeftSingleTRE) / float(throwsTRE)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit left single neighbour during his throw treble in " <<
averageWentLeftSingleTRE << " % of throws." << std::endl; //Displays message
                        float averageWentStraightSingleTRE = (float(wentStraightSingleTRE) / float(throwsTRE)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit straight single neighbour during his throw treble in " <<
averageWentStraightSingleTRE << " % of throws." << std::endl; //Displays message
                        float averageWentRightSingleTRE = (float(wentRightSingleTRE) / float(throwsTRE)) * 100;
                        std::cout << "On average " << name << " missed target he was aiming for and hit right single neighbour during his throw treble in " <<
averageWentRightSingleTRE << " % of throws." << std::endl; //Displays message

                        float totalTreble = averageHitTargetTRE + averageLeftTargetTRE + averageRightTargetTRE + averageWentLeftSingleTRE +
averageWentStraightSingleTRE + averageWentRightSingleTRE;
                        std::cout << "In total " << totalTreble << " % of treble throws." << std::endl; //Displays message
                        std::cout << name << " made " << throwsTRE << " treble throws." << std::endl;
                        std::cout << std::endl;

                        //Innerbull STATS
                        float averageInnerBullIN = (float(innerBullseyesIN) / float(innerBullseyeThrows)) * 100;
                        std::cout << "On average " << name << " hit inner bullseye during his innerbull throw in " << averageInnerBullIN << " % of throws." <<
std::endl; //Displays message
                        float averageOuterBullIN = (float(outerBullseyesIN) / float(innerBullseyeThrows)) * 100;
                        std::cout << "On average " << name << " hit outer bullseye during his innerbull throw in " << averageOuterBullIN << " % of throws." <<
std::endl; //Displays message
                        float averageMissedBullseyesIN = (float(missedBullseyesIN) / float(innerBullseyeThrows)) * 100;
                        std::cout << "On average " << name << " missed bullseye and hit a random number during his innerbull throw in " <<
averageMissedBullseyesIN << " % of throws." << std::endl; //Displays message

                        float totalInner = averageInnerBullIN + averageOuterBullIN + averageMissedBullseyesIN;
                        std::cout << "In total " << totalInner << " % of innerbull throws." << std::endl; //Displays message
                        std::cout << name << " made " << innerBullseyeThrows << " innerbull throws." << std::endl;
                        std::cout << std::endl;

                        //Outerbull STATS
                        float averageInnerBullOUT = (float(innerBullseyesOUT) / float(outerBullseyeThrows)) * 100;
                        std::cout << "On average " << name << " hit inner bullseye during his outerbull throw in " << averageInnerBullOUT << " % of throws." <<
std::endl; //Displays message
                        float averageOuterBullOUT = (float(outerBullseyesOUT) / float(outerBullseyeThrows)) * 100;
                        std::cout << "On average " << name << " hit outer bullseye during his outerbull throw in " << averageOuterBullOUT << " % of throws." <<
std::endl; //Displays message
                        float averageMissedBullseyesOUT = (float(missedBullseyesOUT) / float(outerBullseyeThrows)) * 100;
                        std::cout << "On average " << name << " missed bullseye and hit a random number during his outerbull throw in " <<
averageMissedBullseyesOUT << " % of throws." << std::endl; //Displays message

                        float totalOuter = averageInnerBullOUT + averageOuterBullOUT + averageMissedBullseyesOUT;
                        std::cout << "In total " << totalOuter << " % of outerbull throws." << std::endl; //Displays message
                        std::cout << name << " made " << outerBullseyeThrows << " outerbull throws." << std::endl;
                        std::cout << std::endl;
                        std::cout <<
"==============================================================================================================" << std::endl;
                }
        }


        //float averageBull = (float(innerBullseyes) / float(innerBullseyeThrows)) * 100;
        //std::cout << "On average " << name << " hit inner bullseye in " << averageBull << " % of throws." << std::endl; //Displays message
        //float averageStraightSingle = (float(hitTarget) / float(throws)) * 100;
        //std::cout << "On average " << name << " went straight single in " << averageStraightSingle << " % of throws." << std::endl; //Displays message
        //float averageWentLeft = (float(wentLeftTarget) / float(throws)) * 100;
        //std::cout << "On average " << name << " went left in " << averageWentLeft << " % of throws." << std::endl; //Displays message
        //float averageWentRight = (float(wentRightTarget) / float(throws)) * 100;
        //std::cout << "On average " << name << " went right in " << averageWentRight << " % of throws." << std::endl; //Displays message
        //float averageMissedBullseyes = (float(missedBullseyes) / float(throws)) * 100;
        //std::cout << "On average " << name << " missed bullseye in " << averageMissedBullseyes << " % of throws." << std::endl; //Displays message
        //float total = averageStraightSingle + averageWentLeft + averageWentRight;
        //std::cout << "In total " << total << " % of throws." << std::endl; //Displays message
```

```
		//std::cout << "Had " << error << " errors" << std::endl;
		//std::cout << std::endl;

		Players::resetStats(); //Resets stats so if user back out of stats and play a new one they start anew
}

bool Players::checkWinner()
{
		if (score == 0) {
				//std::cout << name << " has reached score 0 and has won" << std::endl;
				return true;
		}
		else {
				//std::cout << name << " has not yet reached score 0" << std::endl;
				return false;
		}
}

bool Players::validScore(int target, int gamemode, char throwType) {
		int total = score - target; //The score after hitting target
		if (gamemode == 301) { //in 301 it checks if the score after target will be over 50 or 0
				if (total >= 50 || total == 0) {
						//std::cout << "Passes score checks." << std::endl;
						return true;
				}
				else {
						std::cout << "Fails score checks" << std::endl;
						return false;
				}
		}
		else if (gamemode == 501) { //In 501 it checks if the score after hitting target will be bigger then 1 and either be achieved by a double throw or a bull
				if (total > 1) {
						//std::cout << "Passes score checks. Can win with double" << std::endl;
						return true;

				}
				else if (target == 50 && total == 0) {
						//std::cout << "Passes score checks. Can win with bullseye" << std::endl;
						return true;
				}
				else if (throwType == 'd' && total == 0) {
						//std::cout << "Passes score checks. Can win with double" << std::endl;
						return true;
				}
				else if (total <= 1) {
						std::cout << name << "Is BUST" << std::endl;
						score = tempScore;
						return false;
				}
				else {
						std::cout << "error" << std::endl;
				}
		}
		else {
				std::cout << "Error in valid score if else gamemode check" << std::endl;
				return false;
		}
}

void Players::resetStats() {
		//Innerbull
		innerBullseyesIN = 0;
		outerBullseyesIN = 0;
		missedBullseyesIN = 0;
		innerBullseyeThrows = 0;
		//Outerbull
		outerBullseyesOUT = 0;
		innerBullseyesOUT = 0;
		missedBullseyesOUT = 0;
		outerBullseyeThrows = 0;
		//Single
		hitTargetSIN = 0;
		wentLeftTargetSIN = 0;
		wentRightTargetSIN = 0;
		wentLeftDoubleSIN = 0;
		wentStraightDoubleSIN = 0;
		wentRightDoubleSIN = 0;
		wentLeftTrebleSIN = 0;
		wentStraightTrebleSIN = 0;
		wentRightTrebleSIN = 0;
		throwsSIN = 0;
		//Double
		hitTargetDOU = 0;
		wentLeftTargetDOU = 0;
		wentRightTargetDOU = 0;
		missed = 0;
		wentLeftSingleDOU = 0;
		wentStraightSingleDOU = 0;
		wentRightSingleDOU = 0;
		throwsDOU = 0;
		//Treble
		hitTargetTRE = 0;
		wentLeftTargetTRE = 0;
		wentRightTargetTRE = 0;
		wentLeftSingleTRE = 0;
```

```
                wentStraightSingleTRE = 0;
                wentRightSingleTRE = 0;
                throwsTRE = 0;
                winCount = 0;

                //Championships
                championshipsWon = 0;
}

//Increment functions
void Players::wonSet() {
                setsWon++;
}

void Players::wonRound() {
                winCount++;
}

void Players::wonChampionship() {
                championshipsWon++;
}
```

.h

```
#pragma once

#include "DartBoard.h"
#include <string>



class Players
{
private:
        //Arrays
        int oneDart[2][20] =
        {

                {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}, //Row 1
Score
                {2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40} //Row
2 Target
        };

        int twoDart[2][39] =
        {
                {15, 10, 13, 16, 19, 10, 17, 20, 15, 10, 13, 16, 19, 14, 17, 20, 19, 18, 19, 20,
19, 14, 17, 20, 15, 18, 17, 16, 19, 20, 17, 20, 19, 18, 19, 20, 19, 20, 20 }, //Row 1 Target
                {61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100} //Row 2 Score

        };

        int threeDart[2][40] =
        {
                {19, 20, 20, 20, 18, 19, 20, 19, 20, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 19,
20, 17, 18, 19, 20, 25, 19, 20, 18, 19, 20, 20, 20, 20, 20, 20, 20, 19, 20, 19}, //Row 1 Target
                {99, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135,
136, 137, 138, 139} //Row 2 Score
        };




        std::string name;
        //Chances  variables
        int innerbull_chance = 80;
        int outerbull_chance = 80;
        int hit_chance = 80; //Single chance
        int double_chance = 80;
```

```
	int treble_chance = 80;


	//Stats variables
	int	winCount = 0;
	int setsWon = 0;
	int championshipsWon = 0;
	int	score = 0;
	int tempScore = 0;

	//Information variables
	bool infoThrows = false;
	bool infoPercentages = false;
	bool isAhead = false;



	//Stats variables
	//Innerbull
	int innerBullseyesIN = 0;
	int	outerBullseyesIN = 0;
	int	missedBullseyesIN = 0;
	int	innerBullseyeThrows = 0;
	//Outerbull
	int outerBullseyesOUT = 0;
	int	innerBullseyesOUT = 0;
	int	missedBullseyesOUT = 0;
	int	outerBullseyeThrows = 0;
	//Single
	int hitTargetSIN = 0;
	int	wentLeftTargetSIN = 0;
	int	wentRightTargetSIN = 0;
	int	wentLeftDoubleSIN = 0;
	int	wentStraightDoubleSIN = 0;
	int	wentRightDoubleSIN = 0;
	int	wentLeftTrebleSIN = 0;
	int	wentStraightTrebleSIN = 0;
	int	wentRightTrebleSIN = 0;
	int	throwsSIN = 0;
	//Double
	int hitTargetDOU = 0;
	int	wentLeftTargetDOU = 0;
	int	wentRightTargetDOU = 0;
	int	missed = 0;
	int	wentLeftSingleDOU = 0;
	int	wentStraightSingleDOU = 0;
	int	wentRightSingleDOU = 0;
	int	throwsDOU = 0;
	//Treble
	int hitTargetTRE = 0;
	int	wentLeftTargetTRE = 0;
	int	wentRightTargetTRE = 0;
	int	wentLeftSingleTRE = 0;
	int	wentStraightSingleTRE = 0;
	int	wentRightSingleTRE = 0;
	int	throwsTRE = 0;

public:
	Players(std::string, int);
	Players();
	~Players();

	////Getters
	std::string getName();
	//Chance getters
	int getInnerbullChance();
	int getOuterbullChance();
	int getHitChance();
	int getDoubleChance();
	int getTrebleChance();
	//Stats getters
	int getWinCount();
	int getSetsWon();
	int getChampionshipsWon();
```

```
        //Score getters
        int getScore();
        int getTempScore();
        //Info getters
        bool getInfoThrows();
        bool getInfoPercentages();
        bool getIsAhead();

        ///Setters
        void setName(std::string);
        //Chance setters
        void setInnerbullChance(int);
        void setOuterbullChance(int);
        void setHitChance(int);
        void setDoubleChance(int);
        void setTrebleChance(int);
        //Stats setters
        void setWinCount(int);
        void setSetsWon(int);
        void setChampionshipsWon(int);
        //Score setters
        void setScore(int);
        void setTempScore(int);
        //Info setters
        void setInfoThrows(bool);
        void setInfoPercentages(bool);
        void setIsAhead(bool);




        ////Functions
        //Throws
        void throwDart(int, int, char);
        void throwInnerBull(int);
        void throwOuterBull(int);
        void throwSingle(int, int);
        void throwDouble(int, int);
        void throwTreble(int, int);
        //Strategy
        void strategy();

        //Validation
        bool validScore(int, int, char);
        void averageCalculate(int);
        bool checkWinner();
        void resetStats();

        //Increment functions
        void wonSet();
        void wonRound();
        void wonChampionship();



};
```

Code – for each class please copy this page and copy in the code. It does not matter if this goes over one page. Make sure the code is easily readable.

Class Name:
DartBoard

.cpp

```
#include "DartBoard.h"
#include <iostream>
#include <string>

DartBoard::DartBoard()
{
}

DartBoard::~DartBoard()
{
}


int DartBoard::getLeftNeighbour(int target)
{
	return dartboardScores[0][target]; //Returns the left neighbour of target
}

int DartBoard::getRightNeighbour(int target)
{
	return dartboardScores[1][target]; //Returns the right neighbour of target
}
```

.h

```
#pragma once

class DartBoard
{
private:
	//array with the possible score on the dartboard
	static constexpr int dartboardScores[2][21] =
	{
		{0,20,15,17,18,12,13,19,16,14,6,8,9,4,11,10,7,2,1,3,5}, //Row 1 left neighbour
		{0,18,17,19,13,20,10,16,11,12,15,14,5,6,9,2,8,3,4,7,1} //Row 2 right neighbour
	};
public:
	DartBoard();
	~DartBoard();

	//static void getLeftNeighbour(int, int&); //OLD SYSTEM
	//static void getRightNeighbour(int, int&); //OLD SYSTEM

	static int getLeftNeighbour(int);
	static int getRightNeighbour(int);

};
```

Code – for each class please copy this page and copy in the code. It does not matter if this goes over one page. Make sure the code is easily readable.

Class Name:
GameModes

.cpp

```
#include "GameModes.h"
#include <iostream>
#include <string>

GameModes::GameModes()
{
}

GameModes::~GameModes()
{
}


void GameModes::darts301(Players* p1, Players* p2)
{
		p1->setScore(301); //Sets score of both players to 301 for a game of 301
		p2->setScore(301);

		Players* playingPlayer = p1; //Declare a new variable equal to the pointer of p1

		std::cout << "Running automatic 301 darts" << std::endl;

		//Decide who goes first
		std::cout << "Type '1' if you want " << p1->getName() << " to go first or '2' if you want " << p2->getName() << " to go first" << std::endl;
		std::cin >> choice;

		if (choice == 1) { //Changes who goes first
				playingPlayer = p1;
				std::cout << p1->getName() << " will go first" << std::endl;
		}
		else if (choice == 2) {
				playingPlayer = p2;
				std::cout << p2->getName() << " will go first" << std::endl;
		}
		else {
				std::cout << "Invalid input" << std::endl;
		}

		std::cout << "Type how many games you want to simulate: ";
		std::cin >> numberOfGames;

		for (int i = 1; i <= numberOfGames; i++)
		{
				std::cout << "Has ran " << i << " times" << std::endl;

				while (p1->checkWinner() == false && p2->checkWinner() == false) { //Runs until 1 player wins

						int scoreNow = playingPlayer->getScore();
						int difference = scoreNow - 50;

						if (scoreNow > 70 || scoreNow == 50) { //Throws for bull until score gets 50 bull or when score is 50
								playingPlayer->throwDart(50, 301, 'b');
						}
						else if (scoreNow <= 70 && scoreNow > 50) {
								playingPlayer->throwDart(difference, 301, 's');
						}
```

```
            if (playingPlayer == p1) {
                playingPlayer = p2;
            }
            else if (playingPlayer == p2) {
                playingPlayer = p1;
            }
        }
        if (p1->checkWinner() == true) { //Checks who has won
            p1->wonRound();
        }
        else {
            p2->wonRound();
        }
        p1->setScore(301);
        p2->setScore(301);
    }
    std::cout << "================================================" << std::endl;
    std::cout << "  _______  ______  ______  _   _  _    ______  ______ " << std::endl;
    std::cout << "  | ___ \\|  ___|/  ___|| | | || |  |_   _|/  ___|" << std::endl;
    std::cout << "  | |_/ /| |__  \\ `--. | | | || |    | |  \\ `--. " << std::endl;
    std::cout << "  |    / |  __|  `--. \\| | | || |    | |   `--. \\" << std::endl;
    std::cout << "  | |\\ \\ | |___ /\\__/ /| |_| || |_____| |  /\\__/ /" << std::endl;
    std::cout << "  \\_| \\_|\\____/ \\____/  \\___/ \\______/\\_/  \\____/ " << std::endl;
    std::cout << "                                                  " << std::endl;
    std::cout << "================================================" << std::endl;
    float winPercent1 = (float(p1->getWinCount()) / float(numberOfGames)) * 100;
    std::cout << p1->getName() << " won in " << winPercent1 << "% of games" << std::endl;
    float winPercent2 = (float(p2->getWinCount()) / float(numberOfGames)) * 100;
    std::cout << p2->getName() << " won in " << winPercent2 << "% of games" << std::endl;
    std::cout << std::endl;
    p1->averageCalculate(301);
    p2->averageCalculate(301);
    std::cout << "================================================" << std::endl;
    std::cout << "B. Go back to darts 301 menu" << std::endl;

}

void GameModes::darts501(Players* p1, Players* p2) {
    p1->setScore(501);
    p2->setScore(501); //Sets score for both players to 501

    Players* playingPlayer = p1;

    std::cout << "Running automatic 501 darts" << std::endl;

    int random = rand() % 2 + 1; //50/50 who goes first
    if (random == 1) {
        playingPlayer = p1;
        std::cout << p1->getName() << " will go first" << std::endl;
    }
    else {
        playingPlayer = p2;
        std::cout << p2->getName() << " will go first" << std::endl;
    }


    std::cout << "Type how many games you want to simulate: ";
    std::cin >> numberOfGames;

    for (int i = 1; i <= numberOfGames; i++)
    {
        std::cout << "Has ran " << i << " times" << std::endl;
        while (p1->getSetsWon() != 7 && p2->getSetsWon() != 7) //Runs until one player gets to 7 set wins
        {
            while (p1->getWinCount() != 3 && p2->getWinCount() != 3) { //Runs until one player gets to 7 round wins
                while (p1->checkWinner() == false && p2->checkWinner() == false) { //Runs until one player wins round
                    for (int i = 0; i < 3; i++) { //Runs three times to make 3 dart throws
                        if (p1->getScore() > p2->getScore()) { //Checks who is ahead
                            p2->setIsAhead(true);
                            p1->setIsAhead(false);
                        }
                        else {
                            p1->setIsAhead(true);
                            p2->setIsAhead(false);
                        }
                        playingPlayer->strategy(); //Runs strategy
                        if (playingPlayer->checkWinner() == true) { //Stops loop if player wins
                            break;
                        }
                    }

                    if (playingPlayer == p1) { //Switches players
                        playingPlayer = p2;
                    }
                    else {
                        playingPlayer = p1;
                    }

                }
                if (p1->checkWinner() == true) { //Checks who has won round
                    p1->wonRound();
                    //std::cout << p1->getName() << " has won a round." << std::endl; //Commented out to increase speed in
automatic
                }
                else {
                    p2->wonRound();
                    //std::cout << p2->getName() << " has won a round." << std::endl;
                }

                //std::cout << p1->getName() << " has: " << p1->getWinCount() << " round wins" << std::endl; //Commented out to
increase speed in automatic
                //std::cout << p2->getName() << " has: " << p2->getWinCount() << " round wins" << std::endl; //Commented out to
increase speed in automatic
                p1->setScore(501);
                p2->setScore(501);
            }

            if (p1->getWinCount() == 3 ) {
                p1->wonSet();
                //std::cout << p1->getName() << " has won a set." << std::endl; //Commented out to increase speed in automatic
            }
            else if (p2->getWinCount() == 3) {
                p2->wonSet();
                //std::cout << p2->getName() << " has won a set." << std::endl; //Commented out to increase speed in automatic
            }


            //std::cout << p1->getName() << " has: " << p1->getSetsWon() << " set wins" << std::endl; Commented out to increase speed in
automatic
            //std::cout << p2->getName() << " has: " << p2->getSetsWon() << " set wins" << std::endl; Commented out to increase speed in
automatic

            p1->setWinCount(0); //Resets win count for both players so it loops again
            p2->setWinCount(0);
        }
        if (p1->getSetsWon() == 7) { //Checks who won the championsip
            setsWonArray[0][p2->getSetsWon()]++; //Increments the array for player 1 in the collumn where the enemy finished. For example if
enemy finish with 5 sets increment collumn number 5
            p1->wonChampionship(); //Increaments stats
        }
        else if (p2->getSetsWon() == 7) {
            setsWonArray[1][p1->getSetsWon()]++; //Increments the array for player 1 in the collumn where the enemy finished. For example if
enemy finish with 5 sets increment collumn number 5
            p2->wonChampionship(); //Increaments stats
        }

        p1->setSetsWon(0);
        p2->setSetsWon(0);

    }

    std::cout << "================================================" << std::endl;
    std::cout << "  ______  _____  _____  _   _  _    _____  _____ " << std::endl;
    std::cout << "  | ___ \\|  ___|/  ___|| | | || |  |_   _|/  ___|" << std::endl;
    std::cout << "  | |_/ /| |__  \\ `--. | | | || |    | |  \\ `--. " << std::endl;
    std::cout << "  |    / |  __|  `--. \\| | | || |    | |   `--. \\" << std::endl;
    std::cout << "  | |\\ \\ | |___ /\\__/ /| |_| || |____| |  /\\__/ /" << std::endl;
    std::cout << "  \\_| \\_|\\____/ \\____/  \\___/ \\_____/\\_/  \\____/ " << std::endl;
    std::cout << "                                                " << std::endl;
    std::cout << "================================================" << std::endl;

    if (p1->getChampionshipsWon() > p2->getChampionshipsWon()) { //Check who has won more championshops
        std::cout << p1->getName() << " has won more championships. He/she has: " << p1->getChampionshipsWon() << " championship wins" << std::endl;
        float winPercent1 = (float(p1->getChampionshipsWon()) / float(numberOfGames)) * 100;
        std::cout << p1->getName() << " won in " << winPercent1 << "% of championship games" << std::endl;
        std::cout << p2->getName() << " has: " << p2->getChampionshipsWon() << " championship wins" << std::endl;
    }
    else {
        std::cout << p2->getName() << " has won more championships. He/she has: " << p2->getChampionshipsWon() << " championship wins" << std::endl;
        float winPercent2 = (float(p2->getChampionshipsWon()) / float(numberOfGames)) * 100;
        std::cout << p2->getName() << " won in " << winPercent2 << "% of championship games" << std::endl;
        std::cout << p1->getName() << " has: " << p1->getChampionshipsWon() << " championship wins" << std::endl;

    }

    for (int a = 0; a < 2; a++) { //Loops through the array for both players
        if (a == 0) {
            std::cout << p1->getName() << ": " << p2->getName() << std::endl;
        }
        else {
            std::cout << p2->getName() << ": " << p1->getName() << std::endl;
        }
        for (int b = 6; b >= 0; b--) { //Loops through all collumnds in the array
            setsWonTotal[a][b] = (float(setsWonArray[a][b]) / float(numberOfGames)) * 100; //Gets the frequncies of a certain ending happening
            std::cout << "7:" << b << " with "<< setsWonTotal[a][b] << " % chance" << std::endl;
        }
        std::cout << "==========================================" << std::endl;
    }

    p1->averageCalculate(501);
    p2->averageCalculate(501);
}

void GameModes::interactiveDarts501(Players* player) {
    player->setScore(501); //Sets the score of the player to 501 for darts 501

    Players* ai = new Players("Ai", 80);
    ai->setInfoThrows(true); //Turns the information about throws on
    ai->setScore(501); //Sets the score of the ai to 501 for darts 501
    Players* playingPlayer = player;
    int target = 0;
    char trowType = ' ';

    std::cout << "Running interactive 501 darts" << std::endl;

    int random = rand() % 2 + 1; //50/50 chance who goes first
    if (random == 1) {
        playingPlayer = player;
        std::cout << player->getName() << " will go first" << std::endl;
    }
    else {
        playingPlayer = ai;
        std::cout << ai->getName() << " will go first" << std::endl;
    }

    system("Pause"); //Pauses so user can read
    system("CLS");

    while (player->getSetsWon() != 7 && ai->getSetsWon() != 7) //Runs until one player gets to 7 set wins
    {
        while (player->getWinCount() != 3 && ai->getWinCount() != 3) { //Runs until one player gets to 7 round wins
            while (player->checkWinner() == false && ai->checkWinner() == false) { //Runs until one player wins round
                if (playingPlayer == player) { //Checks if it is the user's turn
                    for (int i = 1; i <= 3; i++) { //Runs three times to make 3 dart throws
                        std::cout <<
"=================================================================" << std::endl;
                        std::cout << "Current dart throw " << i << "/3" << std::endl;
                        std::cout << "You current score is " << player->getScore() << "                AI current score is "
<< ai->getScore() << std::endl;
                        std::cout << "You current game won is " << player->getWinCount() << "                AI current
game won is " << ai->getWinCount() << std::endl;
                        std::cout << "You current sets won is " << player->getSetsWon() << "                AI current
sets won is " << ai->getSetsWon() << std::endl;
                        std::cout <<
"=================================================================" << std::endl;
                        std::cout << "Input 's' to make a single throw, 'd' to make a double throw, 't' to make a
treble throw or 'b' to make a throw for bull ";
                        std::cin >> trowType;
                        if (trowType == 's' || trowType == 'd' || trowType == 't') { //Checks if the throw type is
single, double or treble
                            std::cout << "Input what target to aim for (1-20)" << std::endl; //Asks for
target
                            std::cin >> target;
                            std::cout << "-----------------------------------" << std::endl;
                            playingPlayer->throwDart(target, 501, trowType); //Throws dart
                            system("Pause"); //Pauses so player can examine their throw and its result
                            system("CLS");
                            if (playingPlayer->checkWinner() == true) { //Check if the player has won in
order to stop another throw from happening
                                break;
                            }
                        }
                        else if (trowType == 'b') { //Check if user wants to throw for bull
                            std::cout << "Input 25 to aim for outer bull or 50 to aim for inner bull" <<
std::endl;
                            std::cin >> target;
                            std::cout << "---------------------------------------------------------" << std::endl;
                            playingPlayer->throwDart(target, 501, trowType);
                            system("Pause"); //Pauses so player can examine their throw and its result
                            system("CLS");
                            if (playingPlayer->checkWinner() == true) { //Check if the player has won in
order to stop another throw from happening
                                break;
                            }
                        }
                        else {
                            std::cout << "Incorrect input" << std::endl;
                            system("Pause");
                            system("CLS");
                            i--; //Runs loop again if input incorrect
                        }
                    }
                    if (playingPlayer->checkWinner() != true) { //If player has not won switch to ai
                        playingPlayer = ai;
                    }

                }
                else {
                    for (int i = 1; i <= 3; i++) { //Runs three times to make 3 dart throws
                        std::cout << "Current dart throw " << i << "/3" << std::endl;
                        if (player->getScore() > ai->getScore()) { //checks if ahead
                            ai->setIsAhead(true);
                            player->setIsAhead(false);
                        }
                        else {
                            player->setIsAhead(true);
                            ai->setIsAhead(false);
                        }
                        playingPlayer->strategy(); //Runs strategy
                        if (playingPlayer->checkWinner() == true) { //Stops throwing if it wins
                            break;
                        }
                    }
                    system("Pause"); //Pauses so user can examine the results of the throws of the ai
                    system("CLS");
                    if (playingPlayer->checkWinner() != true) {
                        playingPlayer = player;
                    }
                }
            }
            std::cout << playingPlayer->getName() << " has won a game" << std::endl;
            if (player->checkWinner() == true) { //Checks who won
                player->wonRound();
            }
            else {
                ai->wonRound();
            }
            std::cout << "Your current score is " << player->getScore() << "        AI current score is " << ai->getScore() << std::endl;
            std::cout << "Your current game won is " << player->getWinCount() << "        AI current game won is " << ai->getWinCount() <<
std::endl;
            std::cout << "Your current sets won is " << player->getSetsWon() << "        AI current sets won is " << ai->getSetsWon() << std::endl;
            if (playingPlayer == player)
            {
                playingPlayer = ai;
            }
            else {
                playingPlayer = player;
            }
            system("Pause"); //Pauses so user can read
            system("CLS");

            player->setScore(501); //Resets scores to 501 so loops runs again with correct score
            ai->setScore(501); //Resets scores back to 501 so loops runs again with correct score
        }
        if (player->getWinCount() == 3) { //Checks who won set
            player->wonSet();
        }
        else if (ai->getWinCount() == 3) {
            ai->wonSet();
        }

        std::cout << playingPlayer->getName() << " has won a set" << std::endl;
        std::cout << "Your current score is " << player->getScore() << "        AI current score is " << ai->getScore() << std::endl;
        std::cout << "Your current game won is " << player->getWinCount() << "        AI current game won is " << ai->getWinCount() << std::endl;
        std::cout << "Your current sets won is " << player->getSetsWon() << "        AI current sets won is " << ai->getSetsWon() << std::endl;
        system("Pause");
        system("CLS");

        player->setWinCount(0); //sets win count to 0 so loop runs again
        ai->setWinCount(0); //sets win count to 0 so loop runs again
    }
    if (player->getSetsWon() == 7) { //Checks who won championship
        setsWonArray[0][ai->getSetsWon()]++; //Increments the array for player in the collumn where the enemy finished. For example if enemy finish with 5
sets increment collumn number 5
        player->wonChampionship();
    }
    else if (ai->getSetsWon() == 7) {
        setsWonArray[1][player->getSetsWon()]++; //Increments the array for ai in the collumn where the enemy finished. For example if enemy finish with 5
sets increment collumn number 5
        ai->wonChampionship();
    }

    player->setSetsWon(0); //Resets amount of sets won
    ai->setSetsWon(0);

    std::cout << "=================================================" << std::endl;
    std::cout << "  ______  _____  _____  _   _  _     _____  _____ " << std::endl;
    std::cout << "  | ___ \\|  ___|/  ___|| | | || |   |_   _|/  ___|" << std::endl;
    std::cout << "  | |_/ /| |__  \\ `--. | | | || |     | |  \\ `--. " << std::endl;
    std::cout << "  |    / |  __|  `--. \\| | | || |     | |   `--. \\" << std::endl;
    std::cout << "  | |\\ \\ | |___ /\\__/ /| |_| || |_____| |  /\\__/ /" << std::endl;
    std::cout << "  \\_| \\_|\\____/ \\____/  \\___/ \\_____/\\_/  \\____/ " << std::endl;
    std::cout << "                                                  " << std::endl;
    std::cout << "=================================================" << std::endl;

    if (player->getChampionshipsWon() > player->getChampionshipsWon()) {
        std::cout << player->getName() << " has won more championships. He/she has: " << player->getChampionshipsWon() << " championship wins" <<
std::endl;
        std::cout << ai->getName() << " has: " << ai->getChampionshipsWon() << " championship wins" << std::endl;
    }
    else {
        std::cout << ai->getName() << " has won more championships. He/she has: " << ai->getChampionshipsWon() << " championship wins" << std::endl;
        std::cout << player->getName() << " has: " << player->getChampionshipsWon() << " championship wins" << std::endl;
    }

    for (int a = 0; a < 2; a++) { //Loops through the array for both players
        if (a == 0) {
            std::cout << player->getName() << ": " << ai->getName() << std::endl;
        }
        else {
            std::cout << ai->getName() << ": " << player->getName() << std::endl;
        }
        for (int b = 6; b >= 0; b--) { //Loops through all collumnds in the array
            setsWonTotal[a][b] = (float(setsWonArray[a][b]) / float(numberOfGames)) * 100; //Gets the frequncies of a certain ending happening
            std::cout << "7:" << b << " with " << setsWonTotal[a][b] << " % chance" << std::endl;
        }
        std::cout << "===========================================" << std::endl;
    }
```

```
		player->averageCalculate(501);
		ai->averageCalculate(501);

		delete ai; //Deletes object
}
```

.h

```
#pragma once

#include "Players.h"

class GameModes
{
private:
	//Variable declarations
	int choice = 1;
	int numberOfGames = 0;
	bool play = true;
	int setsWonArray[2][7] = {};
	float setsWonTotal[2][7] = {};

public:
	GameModes();
	~GameModes();

	//Gamemode functions
	void darts301(Players*, Players*);
	void darts501(Players*, Players*);
	void interactiveDarts501(Players* player);
};
```

Code – for each class please copy this page and copy in the code. It does not matter if this goes over one page. Make sure the code is easily readable.

Class Name:
Menus

.cpp

```
#include "Menus.h"
#include <iostream>

Menus::Menus()
{
}

Menus::~Menus()
{
}

void Menus::mainMenu() {
        std::cout << "=========================================================" << std::endl;
        std::cout << "  ___  ___      _        ___  ___                 " << std::endl;
        std::cout << "  |  \\/  |     (_)       |  \\/  |                 " << std::endl;
        std::cout << "  | .  . | __ _ _ _ __   | .  . | ___ _ __  _   _ " << std::endl;
        std::cout << "  | |\\/| |/ _` | | '_ \\  | |\\/| |/ _ \\ '_ \\| | | |" << std::endl;
        std::cout << "  | |  | | (_| | | | | | | |  | |  __/ | | | |_| |" << std::endl;
        std::cout << "  \\_|  |_/\\__,_|_|_| |_| \\_|  |_/\\___|_| |_|\\__,_|" << std::endl;
        std::cout << "                                                  " << std::endl;
        std::cout << "=========================================================" << std::endl;
        std::cout << "1. Play" << std::endl;
        std::cout << "2. Settings" << std::endl;
        std::cout << "Q. Quit" << std::endl;
        std::cout << "=========================================================" << std::endl;
}

void Menus::playMenu() {
        std::cout << "==========================" << std::endl;
        std::cout << "  ______ _                " << std::endl;
        std::cout << "  | ___ \\ |               " << std::endl;
        std::cout << "  | |_/ / | __ _ _   _    " << std::endl;
        std::cout << "  |  __/| |/ _` | | | |   " << std::endl;
        std::cout << "  | |   | | (_| | |_| |   " << std::endl;
        std::cout << "  \\_|   |_|\\__,_|\\__, |   " << std::endl;
        std::cout << "                  __/ |   " << std::endl;
        std::cout << "                 |___/    " << std::endl;
        std::cout << "                          " << std::endl;
        std::cout << "==========================" << std::endl;
        std::cout << "1. Play darts 301" << std::endl;
        std::cout << "2. Play darts 501" << std::endl;
        std::cout << "--------------------------" << std::endl;
        std::cout << "B. Back" << std::endl;
        std::cout << "Q. Quit" << std::endl;
        std::cout << "==========================" << std::endl;
}

void Menus::settingsMenu(int player, Players* p1, Players* p2) {
        std::cout << "===============================================" << std::endl;
        std::cout << "   _____      _   _   _                        " << std::endl;
        std::cout << "  /  ___|    | | | | (_)                       " << std::endl;
        std::cout << "  \\ `--.  ___| |_| |_ _ _ __   __ _ ___       " << std::endl;
        std::cout << "   `--. \\/ _ \\ __| __| | '_ \\ / _` / __|" << std::endl;
        std::cout << "  /\\__/ /  __/ |_| |_| | | | | (_| \\__ \\" << std::endl;
        std::cout << "  \\____/ \\___|\\__|\\__|_|_| |_|\\__, |___/" << std::endl;
        std::cout << "                               __/ |           " << std::endl;
        std::cout << "                              |___/            " << std::endl;
        std::cout << "                                               " << std::endl;
        std::cout << "===============================================" << std::endl;
        std::cout << "P. Change which player's stats get changed" << std::endl;
        std::cout << "===============================================" << std::endl;
        if (player == 0) {
                std::cout << "|| " << p1->getName() << " ||" << std::endl;
                std::cout << "1. Change innerbull chance: " << p1->getInnerbullChance() << "%" <<
std::endl;
```

```
        }
        else {
                std::cout << "|| " << p2->getName() << " ||" << std::endl;
                std::cout << "1. Change innerbull chance: " << p2->getInnerbullChance() << "%" <<
std::endl;
        }

        std::cout << "-----------------------------------------------" << std::endl;
        std::cout << "A. Advanced Settings" << std::endl;
        std::cout << "R. Reset to default" << std::endl;
        std::cout << "-----------------------------------------------" << std::endl;
        std::cout << "B. Back" << std::endl;
        std::cout << "Q. Quit" << std::endl;
        std::cout << "===============================================" << std::endl;
}

void Menus::advSettingsMenu(int player, Players* p1, Players* p2) {
        std::cout << "======================================================" << std::endl;
        std::cout << "    ___      _                                   _ " << std::endl;
        std::cout << "   / _ \\     | |                                  | |" << std::endl;
        std::cout << "  / /_\\ \\  __| |__   __ __ _ _ __   ___  ___   __| |" << std::endl;
        std::cout << "  |  _  | / _` |\\ \\ / // _` || '_ \\  / __|/ _ \\ / _` |" << std::endl;
        std::cout << "  | | | || (_| | \\ V /| (_| || | | || (__|  __/| (_| |" << std::endl;
        std::cout << "  \\_| |_/ \\__,_|  \\_/  \\__,_||_| |_| \\___|\\___| \\__,_|" << std::endl;
        std::cout << "                                                " << std::endl;
        std::cout << "======================================================" << std::endl;
        std::cout << "P. Change which player's stats get changed" << std::endl;
        std::cout << "======================================================" << std::endl;
        if (player == 0) {
                std::cout << "|| " << p1->getName() << " ||" << std::endl;
                std::cout << "1. Change outerbull chance: " << p1->getOuterbullChance() << "%" <<
std::endl;
                std::cout << "2. Change single chance: " << p1->getHitChance() << "%" << std::endl;
                std::cout << "3. Change double chance: " << p1->getDoubleChance() << "%" <<
std::endl;
                std::cout << "4. Change treble chance: " << p1->getTrebleChance() << "%" <<
std::endl;
        }
        else {
                std::cout << "|| " << p2->getName() << " ||" << std::endl;
                std::cout << "1. Change outerbull chance: " << p2->getOuterbullChance() << "%" <<
std::endl;
                std::cout << "2. Change single chance: " << p2->getHitChance() << "%" << std::endl;
                std::cout << "3. Change double chance: " << p2->getDoubleChance() << "%" <<
std::endl;
                std::cout << "4. Change treble chance: " << p2->getTrebleChance() << "%" <<
std::endl;
        }
        std::cout << "------------------------------------------------------" << std::endl;
        std::cout << "R. Reset to default" << std::endl;
        std::cout << "B. Back" << std::endl;
        std::cout << "Q. Quit" << std::endl;
        std::cout << "======================================================" << std::endl;
}

void Menus::darts301Menu(bool stats, bool percentages) {
        std::cout << "===================================================" << std::endl;
        std::cout << "  ______            _          _____  _____  __   " << std::endl;
        std::cout << "  |  _  \\            | |        |____ ||  _  |/  | " << std::endl;
        std::cout << "  | | | | __ _  _ __ | |_  ___     / /| |/' |`| | " << std::endl;
        std::cout << "  | | | |/ _` || '__|| __|/ __|     \\ \\|  /| | | | " << std::endl;
        std::cout << "  | |/ /| (_| || |   | |_ \\__ \\ .___/ /\\ |_/ /_| |_" << std::endl;
        std::cout << "  |___/  \\__,_||_|    \\__||___/ \\____/  \\___/ \\___/" << std::endl;
        std::cout << "                                                   " << std::endl;
        std::cout << "===================================================" << std::endl;
        std::cout << "1. Run automatic game" << std::endl;
        std::cout << "2. Display throws that players make during automatic game(It slows down
simulation): ";
        if (stats == true) {
                std::cout << "YES" << std::endl;
        }
        else {
                std::cout << "NO" << std::endl;
        }
        std::cout << "3. Display after game hit percentages: ";
        if (percentages == true) {
                std::cout << "YES" << std::endl;
        }
        else {
                std::cout << "No" << std::endl;
        }
        std::cout << "---------------------------------------------------" << std::endl;
        std::cout << "B. Back" << std::endl;
        std::cout << "Q. Quit" << std::endl;
        std::cout << "===================================================" << std::endl;
}

void Menus::darts501Menu(bool stats, bool percentages) {
        std::cout << "===================================================" << std::endl;
        std::cout << "  ______              _          _____  _____  __  " << std::endl;
        std::cout << "  |  _  \\            | |        |  ___||  _  |/  | " << std::endl;
        std::cout << "  | | | | __ _  _ __ | |_  ___  |___ \\ | |/' |`| | " << std::endl;
        std::cout << "  | | | |/ _` || '__|| __|/ __|     \\ \\|  /| | | | " << std::endl;
        std::cout << "  | |/ /| (_| || |    | |_ \\__ \\ /\\__/ /\\ |_/ /_| |_" << std::endl;
        std::cout << "  |___/  \\__,_||_|    \\__||___/ \\____/  \\___/ \\___/" << std::endl;
        std::cout << "                                               " << std::endl;
        std::cout << "===================================================" << std::endl;
        std::cout << "1. Run automatic game" << std::endl;
        std::cout << "2. Play interactive game" << std::endl;
        std::cout << "3. Display throws that players make during automatic game (It slows down
simulation): ";
        if (stats == true) {
                std::cout << "YES" << std::endl;
        }
        else {
                std::cout << "No" << std::endl;
        }
        std::cout << "4. Display after game hit percentages: ";
        if (percentages == true) {
                std::cout << "YES" << std::endl;
        }
        else {
                std::cout << "No" << std::endl;
        }
        std::cout << "---------------------------------------------------" << std::endl;
        std::cout << "B. Back" << std::endl;
        std::cout << "Q. Quit" << std::endl;
        std::cout << "===================================================" << std::endl;
}
```

.h

```
#pragma once
#include "Players.h"

class Menus
{
private:

public:

        Menus();
        ~Menus();

        static void mainMenu();
        static void playMenu();
        static void settingsMenu(int, Players*, Players*);
        static void advSettingsMenu(int, Players*, Players*);
        static void darts301Menu(bool, bool);
        static void darts501Menu(bool, bool);
};
```

Code – for each class please copy this page and copy in the code. It does not matter if this goes over one page. Make sure the code is easily readable.

Class Name:
Souce

.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <stdlib.h>
#include <conio.h> //Used for input without enter

#include "Players.h"
#include "GameModes.h"
#include "Menus.h"


//Ponter reference to classes

int main() {
    srand(time(0)); //Initiate time for random number

    //Declare variables
    int choice = 0;
    int PlayerEdit = 0;
    int playerChoice = 0;
    int userInput = 0;
    char menuChoice = ' ';
    bool inMain = true;
    bool inPlay = false;
    bool inSettings = false;
    bool inAdvanced = false;
    bool inDarts301 = false;
    bool inDarts501 = false;
    bool darts301Throws = false;
    bool darts301Percentages = false;
    bool darts501Throws = false;
    bool darts501Percentages = false;
    std::string player1Name;
    std::string player2Name;
    int player1Accuracy;
    int player2Accuracy;

    std::cout << "Enter name of player 1: ";
    std::cin >> player1Name;
    std::cout << "Enter innerbull accuracy of player 1: ";
    std::cin >> player1Accuracy;
    std::cout << "Enter name of player 2: ";
    std::cin >> player2Name;
    std::cout << "Enter innerbull accuracy of player 2: ";
    std::cin >> player2Accuracy;

    Players* p1 = new Players(player1Name, player1Accuracy);
    Players* p2 = new Players(player2Name, player2Accuracy);
    GameModes game;


    Menus::mainMenu(); //Displays main menu

    while (menuChoice != 'Q' && menuChoice != 'q') { //Runs until user presses Q to quit
        menuChoice = _getch(); //Gets user input
        switch (menuChoice) {
            case '1':
                if (inMain == true && inPlay == false) { //Runs play menu if in main menu
                    system("CLS"); //Clear console
                    Menus::playMenu(); //Run play menu
                    inMain = false;
                    inPlay = true;
                    break;
                }
                else if (inPlay == true) { //Run Darts 301 menu if in play menu
                    system("CLS"); //Clear console
                    Menus::darts301Menu(darts301Throws, darts301Percentages); //Run darts 301 menu
                    inPlay = false;
                    inDarts301 = true;
                    break;
                }
                else if (inDarts301 == true) { //Run automatic darts 301 game if in darts 301 menu
                    game.darts301(p1, p2); //Runs automatic 301 game
                    break;
                }
                else if (inDarts501 == true) { //Run automatic darts 501 game if in darts 501 menu
                    game.darts501(p1, p2); //Runs automatic 301 game
                    break;
                }
                else if (inSettings == true) { //Change innerbull chance if in settings
                    if (PlayerEdit == 0) { //Check which player is selected
                        std::cout << "Enter new innerbull chance for " << p1->getName() << ": ";
                        std::cin >> userInput;
                        p1->setInnerbullChance(userInput); //Change innerbull chance for player player 1
to user input
                    }
                    else {
                        //Change innerbull chance for player 2
                        std::cout << "Enter new innerbull chance for " << p2->getName() << ": ";
                        std::cin >> userInput;
                        p2->setInnerbullChance(userInput); //Change innerbull chance for player player 21
to user input
                    }
                    system("CLS"); //Clear screen to update
                    Menus::settingsMenu(PlayerEdit, p1, p2); //Display settings menu again with updated chance
                    break;
                }
                else if (inAdvanced == true) { //Change outerbull chance if in advanced menu
                    if (PlayerEdit == 0) { //Check which player is selected
                        std::cout << "Enter new outerbull chance for " << p1->getName() << ": ";
```

```
							std::cin >> userInput;
							p1->setOuterbullChance(userInput); //Change outerbull chance for player player 1 
to user input
						}
						else {
							//Change outerbull chance for player 2
							std::cout << "Enter new outerbull chance for " << p2->getName() << ": ";
							std::cin >> userInput;
							p2->setOuterbullChance(userInput); //Change outerbull chance for player player 1 
to user input
						}
						system("CLS"); //Clear screen to update
						Menus::advSettingsMenu(PlayerEdit, p1, p2); //Display adv settings menu again with updated 
chance
						break;
					}
					break;
				case '2':
					if (inMain == true && inSettings == false) { //Runs settings menu if in main menu
						system("CLS");
						Menus::settingsMenu(PlayerEdit, p1, p2); //Displays settins menu
						inMain = false;
						inSettings = true;
						break;
					}
					else if (inPlay == true){ //Run darts 501 menu if in play menu
						system("CLS");
						Menus::darts501Menu(darts501Throws, darts501Percentages); //Displays darts 501 menu
						inPlay = false;
						inDarts501 = true;
						break;
					}
					else if (inDarts301 == true) { //If in darts391 menu changes the the variable darts301Throws which is used 
to change whether throw information should be displayed
						darts301Throws = !darts301Throws; //Sets darts301Throws to oposite value
						p1->setInfoThrows(darts301Throws); //Uses setter to change it for both players
						p2->setInfoThrows(darts301Throws);
						system("CLS"); //Clears screen
						Menus::darts301Menu(darts301Throws, darts301Percentages); //Loads darts 301 menu again this 
time with updated yes or no
						break;
					}
					else if (inDarts501 == true) { ////If in darts501 run interactive darts
						//RUN INTERACTIVE DARTS
						p1->setInfoThrows(true); //Sets information about throw to ON
						p2->setInfoThrows(true);
						std::cout << "Input 1 to play as " << p1->getName() << " or 2 to play as " << p2->getName() << 
std::endl;
						std::cin >> playerChoice;
						if (playerChoice == 1) { //Depending on player choice gives the function a different player
							game.interactiveDarts501(p1);
						}
						else if (playerChoice == 2) {
							game.interactiveDarts501(p2);
						}
						else {
							std::cout << "Invalid input" << std::endl;
						}
						break;
					}
					else if (inAdvanced == true) { //In in advanced settings menu changes single change
						if (PlayerEdit == 0) {
							//Change single chance player 1
							std::cout << "Enter new single chance " << p1->getName() << ": ";
							std::cin >> userInput;
							p1->setHitChance(userInput);
						}
						else {
							//Change single chance for player 2
							std::cout << "Enter new single chance for " << p2->getName() << ": ";
							std::cin >> userInput;
							p2->setHitChance(userInput);
						}
						system("CLS");
						Menus::advSettingsMenu(PlayerEdit, p1, p2);
						break;
					}
					break;
				case '3':
					if(inDarts301 == true) { ////Change wheter to display DARTS301 DISPLAY PERCENTAGES or not
						darts301Percentages = !darts301Percentages;
						p1->setInfoPercentages(darts301Percentages);
						p2->setInfoPercentages(darts301Percentages);
						system("CLS");
						Menus::darts301Menu(darts301Throws, darts301Percentages);
						break;
					}
					else if (inDarts501 == true) { //Change wheter to display DARTS501 throws information or not
						darts501Throws = !darts501Throws;
						p1->setInfoThrows(darts501Throws);
						p2->setInfoThrows(darts501Throws);
						system("CLS");
						Menus::darts501Menu(darts501Throws, darts501Percentages);
						break;
					}
					else if (inAdvanced == true) { //Changes double chance if in advanced settings
						if (PlayerEdit == 0) {
							//Change double chance player 1
							std::cout << "Enter new double chance for " << p1->getName() << ": ";
							std::cin >> userInput;
							p1->setDoubleChance(userInput);
						}
						else {
							//Change double chance for player 2
							std::cout << "Enter new double chance for " << p2->getName() << ": ";
							std::cin >> userInput;
							p2->setDoubleChance(userInput);
						}
						system("CLS");
						Menus::advSettingsMenu(PlayerEdit, p1, p2);
						break;
					}
					break;
				case '4': //Change wheter to display DARTS501 DISPLAY PERCENTAGES or not
					if (inDarts501 == true) {
						darts501Percentages = !darts501Percentages;
						p1->setInfoPercentages(darts501Percentages);
						p2->setInfoPercentages(darts501Percentages);
						system("CLS");
```

```
                Menus::darts501Menu(darts501Throws, darts501Percentages);
                break;
            }
            else if (inAdvanced == true) { //Changes treble chance if in advanced settings
                if (PlayerEdit == 0) {
                    //Change treble chance player 1
                    std::cout << "Enter new treble chance for " << p1->getName() << ": ";
                    std::cin >> userInput;
                    p1->setTrebleChance(userInput);
                }
                else {
                    //Change double chance for player 2
                    std::cout << "Enter new treble chance for " << p2->getName() << ": ";
                    std::cin >> userInput;
                    p2->setTrebleChance(userInput);
                }
                system("CLS");
                Menus::advSettingsMenu(PlayerEdit, p1, p2);
                break;
            }
            break;
        case 'a':
            if (inSettings == true) { //Enters advanced setting menu if in settings menu
                system("CLS");
                Menus::advSettingsMenu(PlayerEdit, p1, p2);
                inSettings = false;
                inAdvanced = true;
                break;
            }
            break;
        case 'b':
            if (inPlay == true) { //BACK FROM PLAY MENU
                system("CLS");
                Menus::mainMenu();
                inMain = true;
                inPlay = false;
                break;
            }
            else if (inSettings == true) { //BACK FROM SETTINGS MENU
                system("CLS");
                Menus::mainMenu();
                inMain = true;
                inSettings = false;
                break;
            }
            else if (inAdvanced == true) { //BACK FROM ADV SETTINGS MENU
                system("CLS");
                Menus::settingsMenu(PlayerEdit, p1, p2);
                inSettings = true;
                inAdvanced = false;
                break;
            }
            else if (inDarts301 == true) { //BACK FROM DARTS301 MENU
                system("CLS");
                Menus::playMenu();
                inPlay = true;
                inDarts301 = false;
                break;
            }
            else if (inDarts501 == true) { //BACK FROM DARTS501 MENU
                system("CLS");
                Menus::playMenu();
                inPlay = true;
                inDarts501 = false;
                break;
            }
            break;
        case 'r':
            if (inSettings == true) {
                //RESET BASIC SETTINS TO DEFAULT INPUTED BY USER AT START
                if (PlayerEdit == 0) {
                    //Change treble chance player 1
                    p1->setInnerbullChance(player1Accuracy);
                }
                else {
                    p2->setInnerbullChance(player1Accuracy);
                }
                system("CLS");
                Menus::settingsMenu(PlayerEdit, p1, p2);
                break;
            }
            else if (inAdvanced == true) {
                //RESET TO ADVANCED SETTINS TO DEFAULT 80
                if (PlayerEdit == 0) {
                    p1->setOuterbullChance(80);
                    p1->setHitChance(80);
                    p1->setDoubleChance(80);
                    p1->setTrebleChance(80);
                }
                else {
                    p2->setOuterbullChance(80);
                    p2->setHitChance(80);
                    p2->setDoubleChance(80);
                    p2->setTrebleChance(80);
                }
                system("CLS");
                Menus::advSettingsMenu(PlayerEdit, p1, p2);
                break;
            }
            break;

        case 'p':
            if (inSettings == true) { //Changes which player's settings get changed
                PlayerEdit = !PlayerEdit; //sets to oposite
                system("CLS");
                Menus::settingsMenu(PlayerEdit, p1, p2);
            }
            else if (inAdvanced == true) { //Changes which player's adv settings get changed
                PlayerEdit = !PlayerEdit; //sets to oposite
                system("CLS");
                Menus::advSettingsMenu(PlayerEdit, p1, p2);
            }
            break;
        case 'q':
            break;
        default:
            std::cout << "Wrong input" << std::endl;

    }
```

```
		}

		//Deletes objects
		delete p1;
		delete p2;

		return 0;
}
```

.h