Project1

Title

PIG: The Rolling Dice Game_V2

Course

CSC-17A

Class

Fall Semester

Course Code

48983

Due Date

10/25/15

Author

Abdul Hakim Abbas

Table of Contents

Introduction:

This project is based on a rolling dice game called Pig. Pig is a very simple game which was introduced in the Mid 1900's. The game involves a little bit of strategy, but it involves a lot more on your intuition and when to push your luck or to not push it. This has been programed to be played by a human player and a computer player against each other. And it will ask the user for a bet, which then is either will be won by the human player which will double that bet as that player has won the bet. Otherwise the user will lose to the computer player. It involves the player to hold or roll on their two turns, also the computer gets two turns.

How to play:

Pig is played with one die.
Each turn, a player repeatedly rolls a die until either a One is rolled, or the player holds and scores the sum of the rolls (i.e. turn total). At any time a player is faced with two options.

Roll: If the player rolls

- *1: The 1st player(person) scores nothing and it will become the 2nd players(computer) turn.*
- *2 to 6: The number is added to the player's turn total and the player's turn continues.*

Hold: The turn total is added to the player's score and it becomes the next player's turn.

The 1st player to score a 100 or more points wins.

Example of this is that the first player (user) is a roll that is 5. They could hold and score a 5, but choose to roll again. They roll a 2, and can hold with a total of 7 points, but choose to roll again. They roll a 1, and must end their turn without scoring. The next player rolls the sequence 4-5-3-5-5, after which they choose to hold, and it adds their turn total of 22 points to their score.
Since, we don't have a die for this program, the user will simply be typing their action ("1" for roll, and "2" for hold).

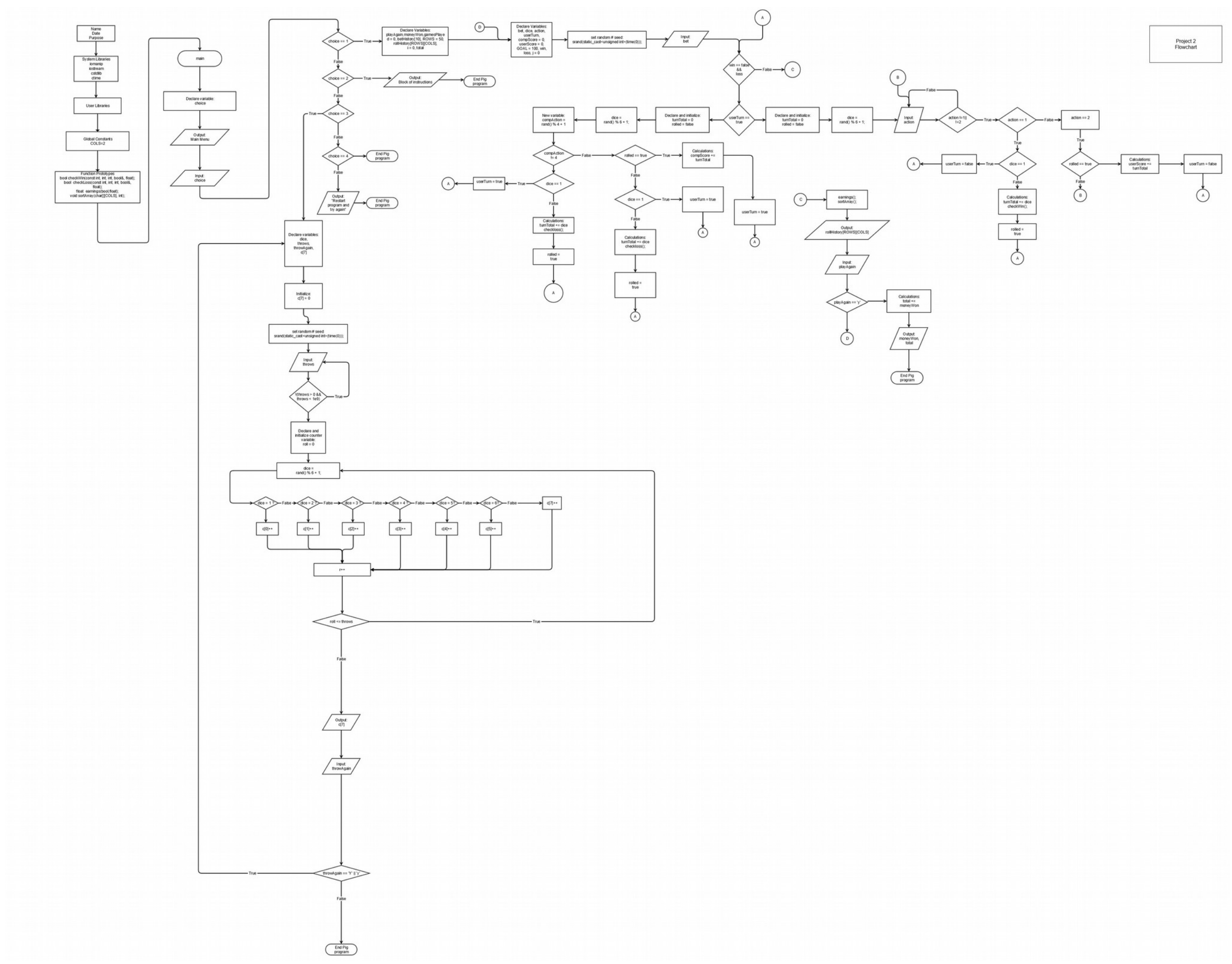
Pseudo-code:

```
Do {

    While (user hasn't won or lost yet) {

        If (user's turn) {

            Do  {

                If (action == 1) {

                    Random dice value

                    Dice value += turnTotal
```

```
                        }
                        else if {action ==2) {
                                userScore += turnTotal
                                if (userScore > 100){
                                        win = true
                                }
                        }
                while (valid input && user's turn && won/loss != true)
        else if (! User's turn) {
                Do {
                        random number 1-4
                        if (random number == 1-3) {
                                random dice value
                                dice value += turnTotal
                        }
                        else if (random number ==4) {
                                compScore += turntTotal
                                if (comScore > 100) {
                                        loss = true
                                }
                        }
                while (! User's turn && won/loss != true)
}
play again? 'y' or 'n'
```

while (play again == 'y')

Flowchart:



Variable List:

Int choice	**Picks 1-4 in main menu**
Char playAgain	Choice variable for playing multiple games
Float moneyWon	will be either positive/negative if win/lose
Int gamesPlayed = 0	keeps track of total games played
Float betHistory[10]	holds all moneyWon values across games
Const int ROWS = 50	Const for maximum estimated number of turns by user

char rollHistory[ROWS] [COLS]	2D array for sorting 'R' -> 'H' and '6' -> '1'
Int I = 0	Counter for bet history
float total	will calculate sum of all moneyWon across games and print on "receipt" file
float bet	Money bet that user gambles each game
int dice	Will hold all values 1-6
short action	Used for "Roll" or "Hold" options
bool userTurn = true	When userTurn is true, it is the user's turn to roll or hold. If false, computer's turn to roll or hold
int compScore = 0 **int userScore = 0**	Overall scores for each game. Turn total added to these after "hold" option is picked
const int GOAL = 100	Game usually ends once player passes GOAL mark
bool win = false **bool lose = false**	If user wins, win = true. Opposite for lose. Either of these turning "true" will end all turns and complete the game
int j = 0	Counter variable for rollHistory[][]
int turnTotal = 0	Added for each dice value != 1. This can potentially be converted to overall score.
bool rolled = false	Keeps track if a roll has occurred for turn. At least one roll required in order to "hold"
char dice	Variable used in to "probability checker"
int throws	Number of loops user wants "p. checker" to do
char throwAgain	Option to run "p. checker" again
ifstream inFile	Controls the file "HowToPlay" text
int wordCount	will help fit block of text inside output parameter
bool swap	Bubble sort main variable
int temp	Holds smaller bubble sort value
ofstream outputFile	Controls the file "receipt.txt"
Fstream binaryFile	Controls the file "test.txt"
Int sides	Number of sides on randomly-sided dice
Int *a	Pointer to unsorted dice throws

Struct Throwresults	float avg; //holds average of dice throws
	 float median; // holds median of dice throws
	 int *mode; //array containing the modes
	 int nModes; //number of modes in the array
	 int maxFreq; //max frequency of modes

Struct User	string name; //user name
	 int age; //age
	 string race; //race

	string gender; //gender
	Day favoriteDay; //Monday through sunday
	bool pass; //if user passed test

Throwresults *pointer	Pointer to Throwresults structure
Throwresults temp	Temporary structure within function
Int *b	Pointer to sorted array
Throwresults **throwresults**	Structure for passing to function and printing results

Resources:

For most of the research to help in coding this game was from Gaddis and I had also utilized websites like Stack overflow, and Git-Hub, and so may other sites.

Reference:

"Pig." Snake Eyes Yard Dice Page « Snake Eyes Yard Dice. N.p., n.d. Web. 20 July 2014.

Checklist:

Chapter 2:
~~int~~ 38
~~float~~ 60
~~bool~~ 54
~~char~~ 92
~~initialization~~ 94
~~arithmetic operators~~ 662
~~comments~~ 551
~~named constants~~ 23
~~strings~~ 340

Chapter 3:
~~cin object~~ 86
~~type casting~~ 80
~~formatting output~~ 332
~~mathematical expressions~~ 503

Chapter 4:
~~relational operators~~ 300
~~if statement~~ 90
~~nested if statement~~ 103
~~switch menu~~ 58

Chapter 5:

~~increment~~ 149
decrement
~~while loop~~ 88
~~counters~~ 491
~~do while loop~~ 96
~~for loop~~ 578

Chapter 6:
~~function prototypes~~ 21
~~pass by value~~ 31
~~returning booleon value~~ 29
~~local variable~~ 72
global variable
~~default arguments~~ 29
~~pass by reference~~ 29
exit() function
stubs and drivers

Chapter 7:
~~array initialization~~ 374
~~2D arrays~~ 61
~~arrays as function arguments~~ 61

Chapter 8:
searching
~~sorting~~ 762

Chapter 9:
~~address operator~~ 793
~~pointers~~ 445
~~pointer notation~~ 707
~~dynamic memory allocation~~ 766
pointer initialization
~~pointers as function parameters~~ 39
~~returning pointers~~ 40
pointers to constants

Chapter 10:
~~Character testing~~ 498
~~Case conversion~~ 453
~~C-strings~~ 496
String member functions

Chapter 11:
~~Structures~~ 32
~~Pointers within~~ structures 35
~~Accessing structure members using dot operators~~ 474
~~Array of structures~~ 465

~~Structures as function argument~~ 42
~~Returning pointer to structure~~ 39
~~Dynamically allocating array of structures~~ 465
~~Enumerated data types~~ 43

Chapter 12:
Passing file stream objects to functions
~~Write to binary file~~ 396
~~Read from binary~~ file 414

Chapter 13 - 16:
~~Abstract class:~~ "Score.h" line 12
~~Base class:~~ "Dice.h" line 12
~~Derived class:~~ "Dice.h" line 17
~~Template class:~~ main.cpp line 63
~~Exception:~~ main.cpp line 120

Additional Information: Line #:

System Level Libraries

iostream	**all**
iomanip	502
cstdlib	80
ctime	80
fstream	324
Cctype	498

Data Types

int	**63**
float	62
short	73
bool	74
const	65
char	61
Enum Day	43

Operators

==	**88**
&&	88
%	97
+=	118

++	149
||	175
<	300
!	382
<=	393
*****	503
&	793
.	474

Copy of the Code:

```
/*
 * File:   main.cpp
 * Author: Abdul-Hakim
 * Purpose: Project Two
 * PIG DICE GAME_V2
 * Created on October 14, 2015, 4:09 PM
 */
//System Level Libraries
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <cctype>
#include <cstring>
#include <string>
using namespace std;

//User Level Libraries
#include "Dice.h"
#include "Score.h"
//Global Constants
const int COLS = 2;
const float initbet = 0.0;
//Function Prototype
void rolling();
void diceArt1();
void diceArt2();
void diceArt3();
void diceArt4();
void diceArt5();
void diceArt6();
void holding();
struct ThrowResults {
    float avg;
    float median;
    int *mode;   //array containing the modes
    int nModes;  //number of modes in the array
    int maxFreq; //max frequency of modes
};
ThrowResults *avgMedMode(int *,int);
int *fillArray(int, int);
int *sortArray(int *, int);
void printStat(ThrowResults);
enum Day {FRIDAY, SATURDAY, SUNDAY, THURSDAY, WEDNESDAY, TUEDSAY, MONDAY};
struct User {
    string name;
    int age;
    string race;
    string gender;
    Day favoriteDay;
    bool pass;
};
bool checkWin(const int, int, int, bool&, float = initbet);
bool checkLoss(const int, int, int, bool&, float = initbet);
void scoreBoard(int, int);
void compScoreBoard(int,int);
void clearScreen(int, int, int);
void compClearScreen(int,int, int);
float earnings(bool, float);
void sortArray(char [][COLS], int);

template <class T>
void totale(T num, T &ttl){
    ttl += num;
}
//Execution Begins Here!
int main(int argc, char** argv) {

    //declare main menu variables
    int choice;
    FancyDice diceIs;
    Score print;
    //main menu output
    cout << " ========================" << endl;
    cout << "||   PIG: THE DICE GAME  ||" << endl;
    cout << "||         PROGRAM       ||" << endl;
    cout << " ========================" << endl << endl << endl;
    cout << "1. Play Pig" << endl;
    cout << "2. How to Play" << endl;
    cout << "3. Dice Probability Checker" << endl;
    cout << "4. Are You Competant Enough to Run This Program?  Found Out! " << endl;
    cout << "5. Quit Program" << endl << endl << endl;

    cout << "Enter choice number and press [enter].";
    cin >> choice;

    //main menu w/ input validation
    switch (choice) {
            //Play the game
        case 1: {
            char playAgain; //for multiple games
            float moneyWon; //will be either positive/negative if win/lose
            int gamesPlayed = 0;    //keeps track of total games played
            float betHistory[10];   //holds all moneyWon values across games
            const int ROWS = 50;    //number of user turns (most likely will not exceed 50)
            char rollHistory[ROWS][COLS];   //array for displaying action ('R' or 'H') and dice value
(1-6)
            int i = 0;  //counter for betHistory
            float total;    //will calculate sum of all moneyWon across games
            do {
                //declare variables
                float bet;  //will be positive value if user wins, negative if lose
                int dice;       //6-sided die
                short action;     //variable for roll or hold
                bool userTurn = true;   //user always gets first turn
                int compScore = 0;      //score of computer
                int userScore = 0;      //score of user
                const int GOAL = 5;   //whoever surpasses this score wins
                bool win = false;  //if true user won, game ends
                bool loss = false;  //if true user lost, game ends
                srand(static_cast<unsigned int>(time(0)));  //set rand() seed
                unsigned int sleep(unsigned int seconds);   //sleep() function setup
                int j = 0;  //counter for rollHistory

                //gamble
                cout << "Type how much money you want to bet: $";
                cin >> bet;


                try {
                    if(bet < 0){
                        throw 1;
                    }
                }


                catch (int e) {
                    cout << "Error number: " << e << endl;
                }


                //Game begins and turns proceed until game has ended
                while (win == false && loss == false) {
                    //when userTurn == true, it is user's turn to roll or hold
                    if (userTurn == true) {
                        //must reset turn total
                        int turnTotal = 0;
                        //must reset roll number
                        bool rolled = false;    //sees if user rolled at least one time (only used for
"hold" input validation)

                        do {
                            dice = rand() % 6 + 1;  //[1,6]
                            cout << endl;
                            cout << "Your turn: Type \"1\" to roll and \"2\" to hold: ";
                            cin >> action;


                            try {
                                if(action < 0){
                                    throw 2;
                                }
                            }

                            catch (int e) {
                                cout << "Error number: " << e << endl;
                            }


                            print.scoreBoard(turnTotal, userScore);
                            //if user picks "roll"
                            if (action == 1) {
                                diceIs.rolling();//rolling();
                                //if dice lands on 1, no value is banked and userTurn becomes false,
meaning computer's turn starts
                                if (dice == 1){
                                    cout << string( 15, '\n' );
                                    cout << "                    |User|" << endl << endl;
                                    cout << "Turn Total: 0                         Score: " <<
userScore << endl;
                                    diceIs.face1();//diceArt1();
                                    userTurn = false;
                                    cout << endl;
                                }
                                //if dice lands on value other than 1, that value is added to turn
total
                                else if (dice == 2) {
                                    print.clearScreen(turnTotal, dice, userScore);
                                    diceIs.face2();//diceArt2();
                                    //turnTotal += dice;
                                    totale(dice, turnTotal);
                                }
                                else if (dice == 3) {
                                    print.clearScreen(turnTotal, dice, userScore);
                                    diceIs.face3();//diceArt3();
                                    totale(dice, turnTotal);

                                }
                                else if (dice == 4) {
                                    print.clearScreen(turnTotal, dice, userScore);
                                    diceIs.face4();//diceArt4();
                                    totale(dice, turnTotal);

                                }
                                else if (dice == 5) {
                                    print.clearScreen(turnTotal, dice, userScore);
                                    diceIs.face5();//diceArt5();
                                    totale(dice, turnTotal);

                                }
                                else if (dice == 6) {
                                    print.clearScreen(turnTotal, dice, userScore);
                                    diceIs.face6();//diceArt6();
                                    totale(dice, turnTotal);

                                }
                                checkWin(GOAL, turnTotal, userScore, win, bet);
                                rolled = true;
                                rollHistory[j][0] = 'R';
                                rollHistory[j][1] = dice + 48;
                                j++;
                            }
                            //if user picks "2"
                            else if (action == 2) {
                                //player must have rolled once before "hold" banks the turn total
                                if (rolled == true) {
                                    diceIs.holding();//holding();
                                    //turn total is banked and added to the overall score of user
                                    userScore += turnTotal;
                                    turnTotal = 0;
                                    print.scoreBoard(turnTotal, userScore);
                                    userTurn = false;
                                    cout << "Banked your total.  " << endl;
                                }
                                //if player has not rolled once before selecting "hold," the program
automatically rolls and similar if else statements from above activate
                                else if (rolled == false) {
                                    cout << "Cannot hold yet." << endl;
                                }
                                rollHistory[j][0] = 'H';
                                rollHistory[j][1] = '-';
                                j++;
                            }
                            else {
                                cout << "Please type a valid action of \"1\" or \"2.\"" << endl;
                            }
                            //user must have picked valid action, it must be the users turn, and the
game must still be active (userScore < GOAL)
                        } while ((action == 1 || action == 2) && userTurn == true && win == false &&
loss == false);
                    }
                    //computer's turn (only input user does is pressing [enter])
                    else if (userTurn == false) {

                        int turnTotal = 0;  //turnTotal is reset for computer
                        bool rolled = false;   //reset roll number

                        cout << "Hit [enter] once to progress through the computer's turn.";
                        cin.ignore(2);

                        do {
                            dice = rand() % 6 + 1;    //same dice and probability as user
                            int compAction = rand() % 4 + 1; //computer will roll 3 out of 4 times
                            print.compScoreBoard(turnTotal, compScore);
                            //roll action should happen 3 out of 4 times
                            if (compAction == 1 || compAction == 2 || compAction == 3) {
                                diceIs.rolling();//rolling();
                                if (dice == 1){
                                    cout << string( 15, '\n' );
                                    cout << "                 |Computer|" << endl << endl;
                                    cout << "Turn Total: 0                         Score: " <<
compScore << endl;

                                    diceIs.face1();//diceArt1();
                                    userTurn = true;
                                    cout << endl;

                                }
                                else if (dice == 2) {
                                    print.compClearScreen(turnTotal, dice, compScore);
                                    diceIs.face3();//diceArt2();
                                    turnTotal += dice;
                                    cout << endl;

                                }
                                else if (dice == 3) {
                                    print.compClearScreen(turnTotal, dice, compScore);
                                    diceIs.face3();//diceArt3();
                                    turnTotal += dice;
                                    cout << endl;
                                }
                                else if (dice == 4) {
                                    print.compClearScreen(turnTotal, dice, compScore);
                                    diceIs.face4();//diceArt4();
                                    turnTotal += dice;
                                    cout << endl;
                                }
                                else if (dice == 5) {
                                    print.compClearScreen(turnTotal, dice, compScore);
                                    diceIs.face5();//diceArt5();
                                    turnTotal += dice;
                                    cout << endl;
                                }
                                else if (dice == 6) {
                                    print.compClearScreen(turnTotal, dice, compScore);
                                    diceIs.face6();//diceArt6();
                                    turnTotal += dice;
                                    cout << endl;
                                }
                                checkLoss(GOAL, turnTotal, compScore, loss, bet);
                                rolled = true;
                            }
                            //hold action should happen 1 out of 4 times
                            else if (compAction == 4) {
                                if (rolled == true) {
                                    diceIs.holding();//holding();
                                    compScore += turnTotal;
                                    turnTotal = 0;
                                    print.compScoreBoard(turnTotal, compScore);
                                    cout << "The computer has banked its turn total." << endl;
                                    userTurn = true;
                                }
                                else if (rolled == false) {
                                    diceIs.rolling();//rolling();
                                    //roll will happen automatically
                                    if (dice == 1){
                                        cout << string( 15, '\n' );
                                        cout << "                 |Computer|" << endl << endl;
                                        cout << "Turn Total: 0                         Score: " <<
compScore << endl;

                                        diceIs.face1();//diceArt1();
                                        userTurn = true;
                                        cout << endl;

                                    }
                                    else if (dice == 2) {
                                        print.compClearScreen(turnTotal, dice, compScore);
                                        diceIs.face2();//diceArt2();
                                        turnTotal += dice;
                                        cout << endl;
                                    }
                                    else if (dice == 3) {
                                        print.compClearScreen(turnTotal, dice, compScore);
                                        diceIs.face3();//diceArt3();
                                        turnTotal += dice;
                                        cout << endl;
                                    }
                                    else if (dice == 4) {
                                        print.compClearScreen(turnTotal, dice, compScore);
                                        diceIs.face4();//diceArt4();
                                        turnTotal += dice;
                                        cout << endl;
                                    }
                                    else if (dice == 5) {
                                        print.compClearScreen(turnTotal, dice, compScore);
                                        diceIs.face5();//diceArt5();
                                        turnTotal += dice;
                                        cout << endl;
                                    }
                                    else if (dice == 6) {
                                        print.compClearScreen(turnTotal, dice, compScore);
                                        diceIs.face6();//diceArt6();
                                        turnTotal += dice;
                                        cout << endl;
                                    }
                                    checkLoss(GOAL, turnTotal, compScore, loss, bet);
                                    rolled = true;
                                }
                                else {
                                    cout << "If you are seeing this, then something horribly wrong has
happened.";
                                }

                            }
                            else {
                                cout << "If you are seeing this, then something horribly wrong has
happened." << endl;
                            }
                            //computer delay between turns
                            for (int x = 0; x < 2; x++) {
                                time_t t1, t2;
                                t1 = time(0);
                                do {
                                    t2 = time(0);
                                } while (difftime (t2, t1) < 1);
                            }
                        } while (userTurn == false && win == false && loss == false);
                    }
                }
                gamesPlayed++;

                cout << endl << "Rolls Stats:" << endl;
                cout << endl;
                sortArray(rollHistory, j);
                moneyWon = earnings(win, bet);
                betHistory[i] = moneyWon;
                i++;

                cout << "Play again?";
                cin >> playAgain;


                try {
                    if(playAgain != 'y' || playAgain != 'n'){
                        throw 3;
                    }
                }


                catch (int e) {
                    cout << "Error number: " << e << endl;
                }


            } while (toupper(playAgain) == 'Y');
            //writing all the money won and lost and total across games
            cout << "Gambling receipt has been printed to a file.";
            ofstream outputFile;
            outputFile.open("receipt.txt");
            for (int j = 0; j < gamesPlayed; j++) {
                outputFile << "Game: " << j + 1 << "       Money won: $ " << setw(8) << betHistory[j]
<< endl;
                total += betHistory[j];
            }
            outputFile << "                          ------------" << endl;
            outputFile << "                            " << setw(9) << total;
            outputFile.close();
        }
            break;

            //How to play
        case 2: {
            //Declare file to be read from
            fstream inFile;
            string word;
            cout << "==========================================================================" <<
endl;
            inFile.open("HowToPlay.txt", ios::in);
            if (inFile) {
                getline(inFile, word);
                while(inFile){
                    cout << word;
                    getline(inFile, word);
                }
                inFile.close();
            }
            else {
                cout << "error";
            }
            cout << endl <<
"==========================================================================" << endl;
            cout << endl;
        }

            break;

            //Dice throw checker (for users who are unsure of the probability of computer dice)
        case 3: {
            cout << endl;

            //Declare Variables
            char dice;
            int throws;
            char throwAgain;
            char throwAgain2;
            fstream binaryFile;
            //Set our random number seed
            srand(static_cast<unsigned int>(time(0)));

            //Repeat program?
            do {
                //Initialize each outcome
                int c[6] = {0,0,0,0,0,0};

                //Input the number of throws
                do {
                    cout << "How many times do you want to throw the 6-sides dice? ";
                    cin >> throws;

                } while (!(throws > 0 && throws < 1e9));

                //Loop the number of times to throw the dice
                for(int roll = 1; roll <= throws; roll++) {
                    //Roll the dice
                    dice = rand() % 6 + 1;    //[1,6]

                    //Determine occurrence of each throw
                    c[dice - 1]++;
                }
                //Write and Read to Binary File
                binaryFile.open("test.txt", ios::out | ios::binary);
                binaryFile.write(reinterpret_cast<char *>(c), sizeof(c));
                binaryFile.close();
                binaryFile.open("test.txt", ios::in | ios::binary);
                binaryFile.read(reinterpret_cast<char *>(c), sizeof(c));
                binaryFile.close();
                //Output the results
                for (int j = 0; j <= 5; j++) {
                    cout << j + 1 << " occurred " << c[j] << " times." << endl;
                }
                cout << "Would you like to roll again? ";
                cout << "Type Y for yes or N for no" << endl;
                cin >> throwAgain;
            } while (toupper(throwAgain) == 'Y');

            do {
                int sides;
                int throws = 0;
                cout << "Input number of sides on the randomly-sided dice: ";
                cin >> sides;
                do {
                    cout << "How many times do you want to throw the randomly-sided dice? ";
                    cin >> throws;

                } while (!(throws > 0 && throws < 1e9));


                int *a = fillArray(throws, sides);
                ThrowResults *pointer = avgMedMode(a, throws);
                ThrowResults throwResults = *pointer;
                printStat(throwResults);
                delete []a;
                cout << "Would you like to roll again? ";
                cout << "Type Y for yes or N for no" << endl;
                cin >> throwAgain2;
            } while (toupper(throwAgain2) == 'Y');

            break;
        }
            //Competence tester
        case 4: {
            int numUsers;
            const int SERIES_SIZE = 3;
            char series[SERIES_SIZE];
```

```
            cout << "How many users will be particpating? :";
            cin >> numUsers;
            User *user;
            user = new User[numUsers];
            for (int i = 0; i < numUsers; i++) {
                bool competent = true;
                bool next = false;
                cin.ignore();
                cout << "Test for user " << i + 1 << ": " << endl;
                cout << "Input your name: ";
                getline(cin, user[i].name);
                cout << "Input your age: ";
                cin >> user[i].age;
                cin.ignore();
                cout << "Input your race: ";
                getline(cin, user[i].race);
                cout << "Input your gender: ";
                cin >> user[i].gender;
                cout << "Thank you. Now to start." << endl;
                cout << "If you can follow simple instructions than you should be able to run this 
program.  Let's give you a little test." << endl;
                while (competent == true && next == false) {
                    cout << "The first test is a simple question: What is your favorite day?  (Enter 
number)" << endl;
                    cout << "1)FRIDAY 2)SATURDAY 3)SUNDAY 4)THURSDAY 5)WEDNESDAY 6)TUEDSAY 7)MONDAY";
                    int x;
                    cin >> x;
                    user[i].favoriteDay = static_cast<Day>(x);
                    if (x > THURSDAY) {
                        competent = false;
                        cout << "No person likes weekdays over the weekend. You have failed." << endl;
                        break;
                    }
                    else {
                        cout << "Good. Now type a series without spaces. Type a symbol, followed by a 
digit, followed by an upper-case letter: ";
                        cin >> series;
                        cin.getline(series, SERIES_SIZE);
                        cout << endl;
                        if(!ispunct(series[0]) && !isdigit(series[1]) && !isupper(series[2])) {
                            competent = false;
                            cout << "You have failed" << endl;
                            break;
                        }
                        else {
                            user[i].pass = true;
                            cout << "Congratulations you have passed the test, you may move onto the 
program." << endl;
                            cout << " ===================================" << endl;
                            cout << "|      Certificate of Competence    |" << endl;
                            cout << "|         " << user[i].name << endl;
                            cout << "|             Great Job!            |" << endl;
                            cout << " ===================================" << endl;
                            next = true;
                        }
                    }

                }
            }
            break;
        }
            //Quit program
        case 5: {
            cout << endl;
            cout << "Program ending." << endl;
        }
            break;
            //Input re-entry
        default: {
            cout << endl;
            cout << "Invalid entry. Restart program and enter (1-4)." << endl;

        }
    }
    return 0;
}
//delay in between rolls
void rolling() {
    time_t t1, t2;
    cout << endl << "rolling." << flush;
    for (int x = 0; x < 3; x++) {
        t1 = time(0);
        do {
            t2 = time(0);
        } while (difftime (t2, t1) < 1);
        cout << "." << flush;
    }
}

void diceArt1() {
    cout << "                    ------- " <<endl;
    cout << "                   |       |" <<endl;
    cout << "                   |   @   |" <<endl;
    cout << "                   |       |" <<endl;
    cout << "                    ------- " <<endl << endl;
    cout << string( 20, '\n' );
}
void diceArt2() {
    cout << "                    ------- " <<endl;
    cout << "                   |       |" <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                   |       |" <<endl;
    cout << "                    ------- " <<endl << endl;
    cout << string( 20, '\n' );
}
void diceArt3() {
    cout << "                    ------- " <<endl;
    cout << "                   |     @ |" <<endl;
    cout << "                   |   @   |" <<endl;
    cout << "                   | @     |" <<endl;
    cout << "                    ------- " <<endl << endl;
    cout << string( 20, '\n' );
}
void diceArt4() {
    cout << "                    ------- " <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                   |       |" <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                    ------- " <<endl << endl;
    cout << string( 20, '\n' );

}
void diceArt5() {
    cout << "                    ------- " <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                   |   @   |" <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                    ------- " <<endl << endl;
    cout << string( 20, '\n' );
}
void diceArt6() {
    cout << "                    ------- " <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                   | @   @ |" <<endl;
    cout << "                    ------- " <<endl << endl;
    cout << string( 20, '\n' );
}
//just like "rolling" delay
void holding() {
    time_t t1, t2;
    cout << endl << "holding." << flush;
    for (int x = 0; x < 3; x++) {
        t1 = time(0);
        do {
            t2 = time(0);
        } while (difftime (t2, t1) < 1);
        cout << "." << flush;
    }
}
bool checkWin(const int pointCap, int potentialScore, int overallScore, bool& win, float wager) {
    if (potentialScore + overallScore > pointCap) {
        //set display for monetary value
        cout << setprecision(2) << fixed << showpoint;
        cout << "You win! You take the pot and walk with $" << wager * 2 << " dollars." << endl;
        win = true;
        return win;
    }
    else {
        win = false;
        return win;
    }
}
bool checkLoss(const int pointCap, int potentialScore, int overallScore, bool& loss, float wager) {
    if (potentialScore + overallScore > pointCap) {
        //set display for monetary value
        cout << setprecision(2) << fixed << showpoint;
        cout << "The computer beat you!  You lose $" << wager << " dollars." << endl;
        loss = true;
        return loss;
    }
    else {
        loss = false;
        return loss;
    }
}
//scoreBoard, compScoreBoard, clearScreen, compclearScreen and the diceArts all make sure
//screen is new and updated for each turn
void scoreBoard(int potentialScore, int overallScore) {
    cout << string( 32, '\n' );
    cout << "                    |User|" << endl << endl;
    cout << "Turn Total: " << potentialScore << "                        Score: " << overallScore <<
endl;
    cout << string( 26, '\n' );
}
void compScoreBoard(int potentialScore, int compOverallScore) {
    cout << string( 32, '\n' );
    cout << "                  |Computer|" << endl << endl;
    cout << "Turn Total: " << potentialScore << "                        Score: " << compOverallScore
<< endl;
    cout << string( 26, '\n' );
}
void clearScreen(int potenitalScore, int diceScore, int overallScore) {
    cout << string( 15, '\n' );
    cout << "                    |User|" << endl << endl;
    cout << "Turn Total: " << potenitalScore + diceScore << "                        Score: " <<
overallScore << endl;
}
void compClearScreen(int potenitalScore, int diceScore, int compOverallScore){
    cout << string( 15, '\n' );
    cout << "                  |Computer|" << endl << endl;
    cout << "Turn Total: " << potenitalScore + diceScore << "                        Score: " <<
compOverallScore << endl;
```

```
}
float earnings(bool win, float initialBet) {
    float moneyWon;
    if (win) {
        moneyWon = initialBet;
    }
    else {
        moneyWon = (-initialBet);
    }
    return moneyWon;
}
//sorts the 2D array, descending order '6' -> '-' and 'R' -> 'H'
void sortArray(char rollHistory[][COLS], int turns) {
    bool swap;
    int temp;
    do {
        swap = false;
        for (int i = 0; i < turns; i++) {
            if (rollHistory[i][0] < rollHistory[i + 1][0]) {
                temp = rollHistory[i][0];
                rollHistory[i][0] = rollHistory[i + 1][0];
                rollHistory[i + 1][0] = temp;
                swap = true;
            }
        }
    } while (swap);
    do {
        swap = false;
        for (int i = 0; i < turns; i++) {
            if (rollHistory[i][1] < rollHistory[i + 1][1]) {
                temp = rollHistory[i][1];
                rollHistory[i][1] = rollHistory[i + 1][1];
                rollHistory[i + 1][1] = temp;
                swap = true;
            }
        }
    } while (swap);
    cout << "Action         Roll" << endl;
    for (int k = 0; k <= turns; k++) {
        cout << "  " << rollHistory[k][0] << "              " << rollHistory[k][1] << endl;

    }
}

int *fillArray(int throws, int sides) {
    int *d = new int[throws];
    //Loop the number of times to throw the dice
    for(int roll = 0; roll < throws; roll++) {
        *(d + roll) = rand() % sides + 1;
    }
    for(int roll = 0; roll < throws; roll++) {
        cout << *(d + roll) << endl;
    }
    return d;
}
int *sortArray(int *array, int size) {
    //Allocate a sortable array
    int *b = new int[size];
    //Copy the array
    for(int i = 0; i < size; i++){
        b[i] =  array[i];
    }
    //Sort the array
    for(int i = 0;i < size - 1; i++){
        for(int j = i + 1;j < size; j++){
            if(b[i] > b[j]){
                b[i] = b[i]^b[j];
                b[j] = b[i]^b[j];
                b[i] = b[i]^b[j];
            }
        }
    }
    cout << "Sorted Array: " << endl;
    for (int j = 0; j < size; j++) {
        cout << b[j] << " ";
    }
    cout << endl;
    return b;
}
ThrowResults *avgMedMode(int *a,int n) {
    ThrowResults temp;
    //Create a parallel array to sort
    int *b=sortArray(a,n);
    //Count to max frequency
    int count=0,maxFreq=0;
    for(int i=1;i<n;i++){
        if(b[i]==b[i-1]){
            count++;
            if(maxFreq<count)maxFreq=count;
        }else{
            count=0;
        }
    }
    temp.maxFreq = maxFreq+1;
    //Count number of modes
    count=0;
    int nmodes=0;
    for(int i=1;i<n;i++){
        if(b[i]==b[i-1]){
            count++;
            if(maxFreq==count)nmodes++;
        }else{
            count=0;
        }
    }
    temp.nModes = nmodes;
    //Declare and fill the mode array
    int *mode=new int[nmodes];
    nmodes=0;
    count=0;
    for(int i=1;i<n;i++){
        if(b[i]==b[i-1]){
            count++;
            if(maxFreq==count)mode[nmodes++]=b[i];
        }else{
            count=0;
        }
    }
    temp.mode = mode;
    //avg
    int total = 0;
    for (int i = 0; i < n; i++) {
        total += b[i];
    }
    temp.avg = static_cast<float>(total)/static_cast<float>(n);
    //median
    if ((n % 2) != 0) {
        temp.median = b[((n - 1) / 2)];
    }
    else if ((n % 2) == 0) {
        temp.median = (static_cast<float>(b[(n/2) - 1]) + static_cast<float>(b[(n / 2)]))/2.0;
    }
    //Clean up and return
    delete []b;
    ThrowResults *pointer = &temp;
    return pointer;
```

```
}
void printStat(ThrowResults throwResults) {
    cout << "Average: " << throwResults.avg << endl;
    cout << "Median: " << throwResults.median << endl;
    cout << "Number of Modes: " << throwResults.nModes << endl;
    cout << "Maximum Frequency: " << throwResults.maxFreq << endl;
    for (int i = 0; i < throwResults.nModes; i++){
        cout << "Mode: " << throwResults.mode[i] << " ";
    }
    cout << endl;
}
```