**Machine Project**

**Miranda, Juan Miguel – S22A**

**Link to Flowchart:** [**https://www.figma.com/file/IDduFwZf2qHmHJWoC0EQCV/MP-1-Flow?type=whiteboard&node-id=0-1&t=sPMaEXh4Zi0Dt8uq-0**](https://www.figma.com/file/IDduFwZf2qHmHJWoC0EQCV/MP-1-Flow?type=whiteboard&node-id=0-1&t=sPMaEXh4Zi0Dt8uq-0)

**Disclaimer:**

**About 80% of the code was made under the initial assumption that arrays are not allowed. This resulted in many variable assigning to get categories grouped together, etc. which made the code very long and tedious in some places. I hope you can forgive that since it initially wasn’t allowed. The code can be vastly improved with arrays and especially with objects.**

**I thought that that was the point of the MP which showed that you mastered the basics so it was limited like that which turned out to not have been the case. I hope I don’t get punished for this misunderstanding. This was the harder route that I wish I didn’t take. It was emphasized a lot in the beginning that using of arrays and non-discussed topics would lead to 0.**

**I hope there aren’t deductions for the way I will attempt to explain the functions and tests such as setting certain functions as the topic functions which just means where the function where the flow being discussed is occuring. Although certain cases effects aren’t necessarily “outputs” in the terminal, they will be talked about in Expected Outputs which will just mean Expected Effects.**

**The test cases mainly focuses on a 3-player game, smaller pieces from various other games will be shown to not restate the same outcomes with minor differences.**

**Topic function main()**

Function: void PrintJeopardy()

Function Description: Prints Jeopardy logo

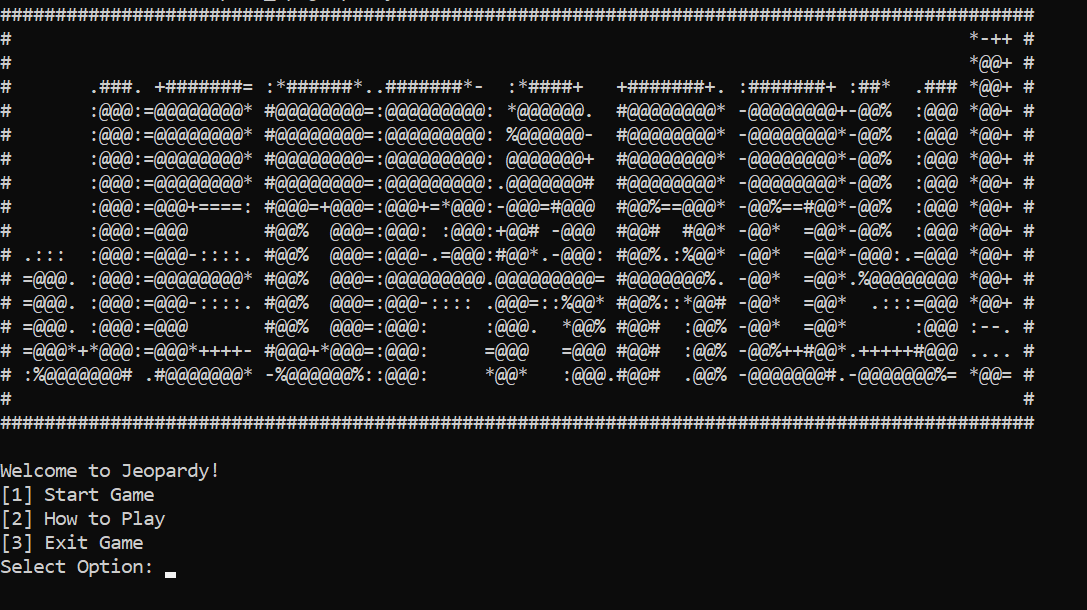
**A black background with white text

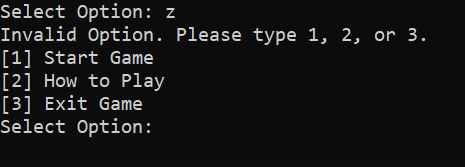
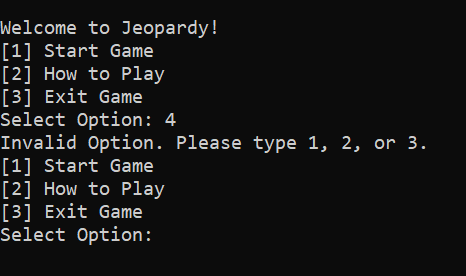
Description automatically generated**

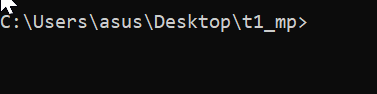
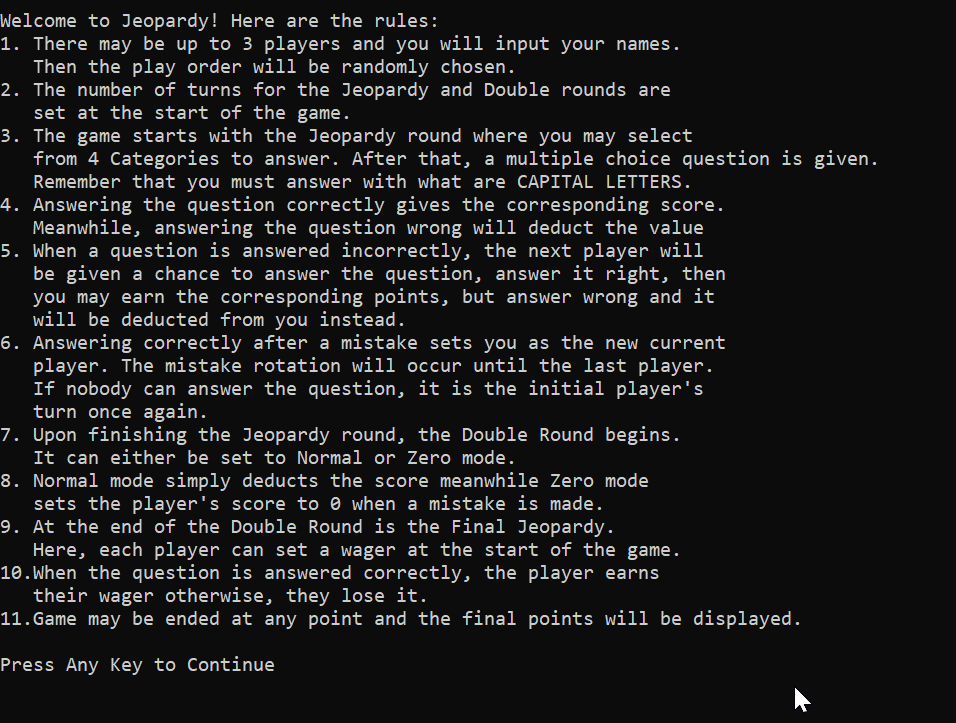
Function: char StartMenu()

Function Description: Displays the starting menu and the possible menu option. Asks for a char input, returns the first digit of the input. Validates the input and asks for a reentry when the first digit of the input is not 1, 2, or 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| An invalid input | 4 | Invalid Option. Please type 1, 2, or 3. | Invalid Option. Please type 1, 2, or 3. | Passed |
| An invalid input | z | Invalid Option. Please type 1, 2, or 3. | Invalid Option. Please type 1, 2, or 3. | Passed |
| A valid input, triggers the game | 1 | Moves to the next section that sets details for upcoming game | Moves to the next section that sets details for upcoming game | Passed |
| A valid input, triggers the instructions | 2 | Displays instructions list | Displays instructions list | Passed |
| A valid input, exits the program | 3 | Exits Program | Exits Program | Passed |

****

****

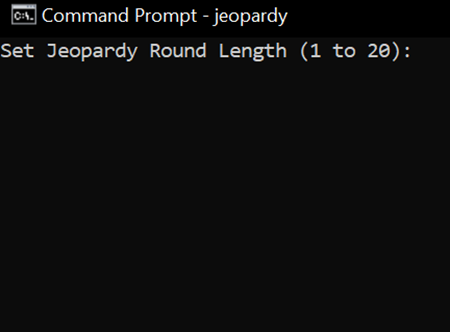
****

**Case 1 – Start Game Contents**

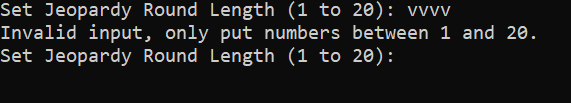
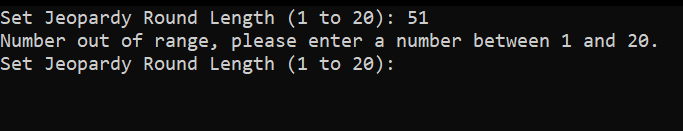
Function: void ClearTerminal()

Function Description: uses system(“cls”); to clear the terminal

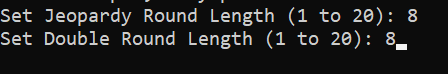
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| none | none | Cleared screen | Cleared screen | Passed |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A number below 0 or beyond 20 | 51 | Number out of range, please enter a number between 1 and 20. | Number out of range, please enter a number between 1 and 20. | Passed |
| Not an integer | vvvv | Number out of range, please enter a number between 1 and 20. | Number out of range, please enter a number between 1 and 20. | Passed |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A valid input | 8 | Moves on to double round length | Moves on to double round length | Passed |
| A valid input | 8 | Moves on to double round penalty setup | Moves on to double round length | Passed |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Choice for normal penalty mode in Double round | 1 | Chosen Penalty Mode: 1 Moves on to player count setup | Chosen Penalty Mode: 1  Moves on to player count setup | Passed |

A black background with white text

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Player count that isn’t 1, 2, or 3 | 5 | Invalid player count. Please type 1, 2, or 3. | Invalid player count. Please type 1, 2, or 3. | Passed |
| Player count that isn’t 1, 2, or 3 | assd | Invalid player count. Please type 1, 2, or 3. | Invalid player count. Please type 1, 2, or 3. | Passed |

**-End of main being the topic function-**

**Topic Function: PlayGame()**

Function: void PlayGame(int nLength1, int\* nLength2, int\* nPlayersHold, int\* nTurnTrackerHold,

char\* strFirstHold, char\* strSecondHold, char\* strThirdHold, char\* strActivePlayerHold,

int\* nFirstScoreHold, int\* nSecondScoreHold, int\* nThirdScoreHold, int\* nActiveScoreHold, int nDoubleMode)

Function Description: Contains the functions and flow to play the jeopardy round. Won’t have outputs itself but contains the functions within that make progress

Function: int GetPlayers()

Function Description: Asks for the player count then validates it. Makes input from char to int

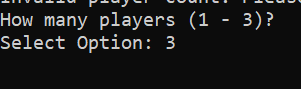
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Invalid number of players | 5 | Invalid player count. Please type 1, 2, or 3. | Invalid player count. Please type 1, 2, or 3. | Passed |
| Invalid number of players | assd | Invalid player count. Please type 1, 2, or 3. | Invalid player count. Please type 1, 2, or 3. | Passed |

**A black screen with white text

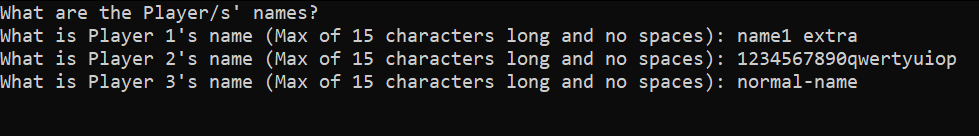
Description automatically generatedA black screen with white text

Description automatically generated**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A valid input | 3 | Moved onto inputting of 3 names | Moved onto inputting of 3 names | Passed |

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Becomes 1 word because of input buffer | name1 extra | Becomes “name1” | Becomes “name1” | Passed |
| Becomes 15 digits because of input buffer | 1234567890qwertyuiop | Becomes “1234567890qwert” | Becomes “1234567890qwert” | Passed |
| Stays the same, a very valid name | normal-name | Stays as “normal-name” | Stays as “normal-name” | Passed |

****

A black screen with white text

Description automatically generated

Function: void OrderPlayers(int\* nPlayers, char\* strPlayer1Name, char\* strPlayer2Name, char\* strPlayer3Name, char\* strFirst, char\* strSecond, char\* strThird)

Function Description: Using the current time as the seed, roll 3 random numbers to assign the order to the players. Keeps rerolling if a duplicate number is gotten. Assigns the players’ names and scores to specific variables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Resulting in:  2  3  1 | name1  1234567890qwert  normal-name | “The Play Order Will Be:”  A permutation of the 3 names when randomly selected | “The Play Order Will Be:”  A permutation of the 3 names when randomly selected | Passed |

A black screen with white text

Description automatically generated

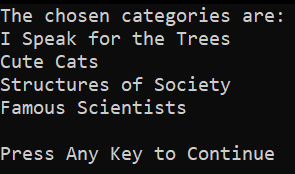
**Category Selection**

In order to save time, only the first category’s contents will be listed in the test cases. The same process applies to all other categories but the others include number rerolling when a duplicate number happens.

Function: int RandMinMax(int min, int max)

Function Description: Returns a random number between the min and max (inclusive)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Resulted in 7 | In the background: nCateg1 is assigned | Assigned question contents | Assigned question contents | Passed |



**GetContent Functions**

In order to save time, only the first of the 3 will be described in the test cases. The same process applies but with different datatypes and attributes.

Function: void GetContent1(int TitleNum, char\* strTitle, char\* CatQ1, char\* CatQ2, char\* CatQ3, char\* CatQ4, char\* CatQ5)

Function Description: Within the specified variables, assigns the selected random number’s item. In this case, assigns the details of the 7th category through “=” and strcpy()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| With the random number being 7, the properties of the 7th category in questions1.c was assigned which is the category about trees. | none | strTitle = “I Speak for the Trees”  Cat1Q1 = "Which tree is known for its vibrant red and orange leaves in the autumn?"  Cat1Q2 = "What is the name of a tree that produces conkers, which are often used in games?"  Cat1Q3 = "Which tree is famous for its white bark and can be found in North America?"  Cat1Q4 = "What is the tallest type of tree in the world?"  Cat1Q5 = "Which tree species is commonly associated with the production of paper?" | strTitle = “I Speak for the Trees”  Cat1Q1 = "Which tree is known for its vibrant red and orange leaves in the autumn?"  Cat1Q2 = "What is the name of a tree that produces conkers, which are often used in games?"  Cat1Q3 = "Which tree is famous for its white bark and can be found in North America?"  Cat1Q4 = "What is the tallest type of tree in the world?"  Cat1Q5 = "Which tree species is commonly associated with the production of paper?" | Passed |

A black screen with white text

Description automatically generated

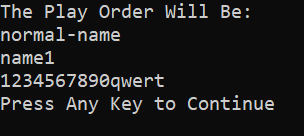
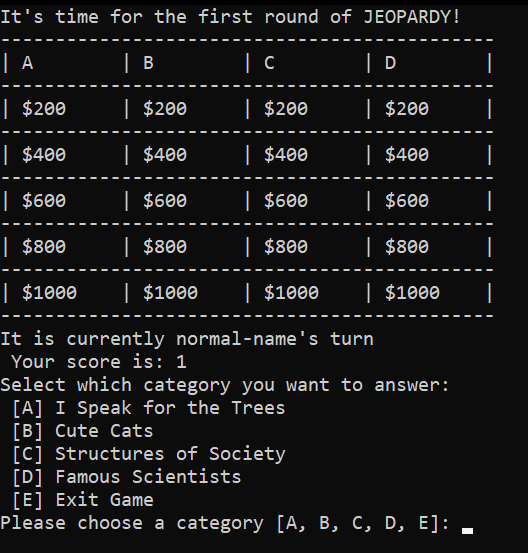
Function: void SetPlayer(int nTurnTracker, char\* Player1Name, char\* Player2Name, char\* Player3Name,

int Player1Score, int Player2Score, int Player3Score, char\* strActivePlayer, int\* nActiveScore)

Function Description: Based on the nTurnTracker, uses switch cases to assign strActivePlayer for the name and nActiveScore for the current player’s score

**Note: Player1 is given a score at the start of 1, 2 is given 2, and 3 is given 3. This allows for easier identification while debugging and for test case demonstration. This will not be the case for the final output.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| The player with the assigned number 1 is the first player | nTurnTracker = 1 | strActivePlayer = “normal-name”  nActiveScore = 1 | strActivePlayer = “normal-name”  nActiveScore = 1 | Passed |

****

Function:

**-End of PlayGame being the topic function-**

Function: void PrintTable(int row1a, int row1b, int row1c, int row1d,

int row2a, int row2b, int row2c, int row2d,

int row3a, int row3b, int row3c, int row3d,

int row4a, int row4b, int row4c, int row4d,

int row5a, int row5b, int row5c, int row5d)

Function Description: Is given the value of all prizes and puts them in a table. Uses the functions PrintLine() and PrintRow() which just makes the layout. All of which can be inferred from the image they produce.

**A screenshot of a game

Description automatically generated**

**Topic Function: PickChoice**

Function: void PickChoice(

char\* Title1, char\* Title2, char\* Title3, char\* Title4,

char\* Cat1Q1, char\* Cat2Q1, char\* Cat3Q1, char\* Cat4Q1,

char\* Cat1Q2, char\* Cat2Q2, char\* Cat3Q2, char\* Cat4Q2,

char\* Cat1Q3, char\* Cat2Q3, char\* Cat3Q3, char\* Cat4Q3,

char\* Cat1Q4, char\* Cat2Q4, char\* Cat3Q4, char\* Cat4Q4,

char\* Cat1Q5, char\* Cat2Q5, char\* Cat3Q5, char\* Cat4Q5,

char\* Cat1Ch1, char\* Cat2Ch1, char\* Cat3Ch1, char\* Cat4Ch1,

char\* Cat1Ch2, char\* Cat2Ch2, char\* Cat3Ch2, char\* Cat4Ch2,

char\* Cat1Ch3, char\* Cat2Ch3, char\* Cat3Ch3, char\* Cat4Ch3,

char\* Cat1Ch4, char\* Cat2Ch4, char\* Cat3Ch4, char\* Cat4Ch4,

char\* Cat1Ch5, char\* Cat2Ch5, char\* Cat3Ch5, char\* Cat4Ch5,

char Cat1Ans1, char Cat2Ans1, char Cat3Ans1, char Cat4Ans1,

char Cat1Ans2, char Cat2Ans2, char Cat3Ans2, char Cat4Ans2,

char Cat1Ans3, char Cat2Ans3, char Cat3Ans3, char Cat4Ans3,

char Cat1Ans4, char Cat2Ans4, char Cat3Ans4, char Cat4Ans4,

char Cat1Ans5, char Cat2Ans5, char Cat3Ans5, char Cat4Ans5,

int\* Cat1P1, int\* Cat2P1, int\* Cat3P1, int\* Cat4P1,

int\* Cat1P2, int\* Cat2P2, int\* Cat3P2, int\* Cat4P2,

int\* Cat1P3, int\* Cat2P3, int\* Cat3P3, int\* Cat4P3,

int\* Cat1P4, int\* Cat2P4, int\* Cat3P4, int\* Cat4P4,

int\* Cat1P5, int\* Cat2P5, int\* Cat3P5, int\* Cat4P5,

int\* nTurnTracker, int nPlayers,

char\* strFirst, char\* strSecond, char\* strThird,

int\* nFirstScore, int\* nSecondScore, int\* nThirdScore,

char\* strActivePlayer, int\* nActiveScore,

int\* CatAProgress, int\* CatBProgress, int\* CatCProgress, int\* CatDProgress, int\* CatEProgress, int\* nJeopardyProgress, int\* nLength2,

int nDoubleMode

)

Function Description: Is looped based on Jeopardy Round length which becomes nJeopardyProgress. Contains all of the functions to answer items in the board and select questions. The bulk of the actions occur within.

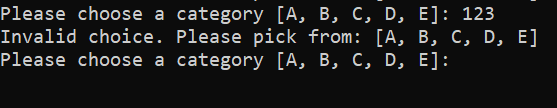
**You can consider this the container that contains all of the other functions without necessarily outputting anything itself. As such, I won’t put a table for this one as a whole but instead for it’s contents.**

Choose Category

Description: Asks the player to input a category to answer. Will validate if a choice is valid. But if the category has been completed, it will stop it from being chosen. Just reads the first digit given, input buffers the rest.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| An invalid input | F | “Invalid choice. Please pick from: [A, B, C, D, E]: ” | “Invalid choice. Please pick from: [A, B, C, D, E]: ” | Passed |
| An invalid input | 123 | “Invalid choice. Please pick from: [A, B, C, D, E]: ” | “Invalid choice. Please pick from: [A, B, C, D, E]: ” | Passed |
| The question hasn’t been completed | A | Proceeds to choose question | Proceeds to choose question | Passed |
| The question has been completed | A | The category has been completed. Pick another. | The category has been completed. Pick another. | Passed |

A black background with white text

Description automatically generatedA black screen with white text

Description automatically generated

**A screenshot of a computer screen

Description automatically generated**

E to exit

Choosing the option E will set the nJeopardyProgress to 0. The nJeopardyProgress is set to the length of the round and gets deducted as progress happens. When it is 0, it exits the loop resultingin the exit and return to the main menu.

A black screen with white text

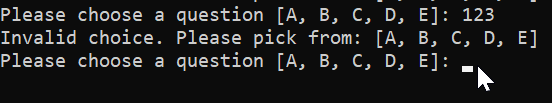
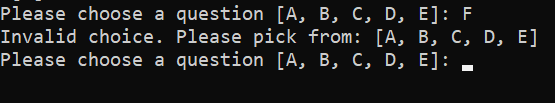
Description automatically generatedA screenshot of a computer

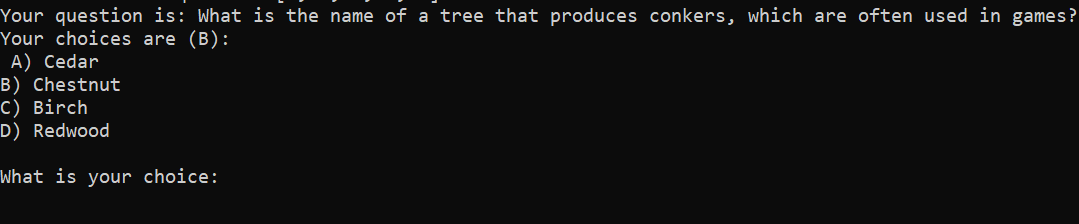
Description automatically generated

Function: int ValidateQuestion(int\* P1, int\* P2, int\* P3, int\* P4, int\* P5)

Function Description: Asks the player for the question that they want to answer and checks for validity, approves it when the question’s price value is greater than 0. Otherwise, asks for a different input.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Invalid input | F | Invalid choice. Please pick from [A, B, C, D, E]: | Invalid choice. Please pick from [A, B, C, D, E]: | Passed |
| Invalid input | 123 | Invalid choice. Please pick from [A, B, C, D, E]: | Invalid choice. Please pick from [A, B, C, D, E]: | Passed |
| The question has not been completed | B B B | Proceeds to choose answer | Proceeds to choose answer | Passed |
| The question has been completed | B | That question has already been answered. | That question has already been answered. | Passed |

****A black background with white text

Description automatically generated****

Function: void PickAnswer (char\* CatQ, char\* CatCh, char CatAns, int\* CatP,

int\* nTurnTracker, int nPlayers, int nFailCounter,

char\* strFirst, char\* strSecond, char\* strThird,

int\* nFirstScore, int\* nSecondScore, int\* nThirdScore,

char\* strActivePlayer, int\* nActiveScore, int nDoubleMode)

Function Description: Is the main container of the functions that will check whether a player’s answer is right or wrong.

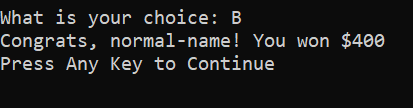
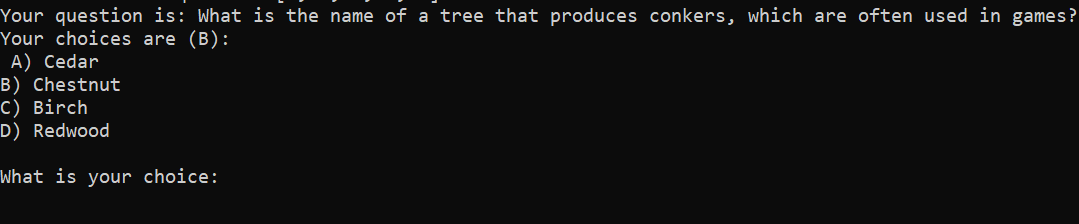
Starts with:

Function: int CheckAnswer(char\* CatQ, char\* CatCh, char CatAns, int\* nActiveScore, int\* CatP, char\* strActivePlayer, int nDoubleMode)

Function Description: Will see if the input matches the question’s answer. If it does, it adds the points to the nActiveScore. If not, it will deduct points or in the case of Double round, either deduct as normal or set it to 0. Sets the value of the prize to 0 to make it unanswerable. Wrong answers lead to ChangePlayers.

**Correct Answer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A correct answer is given | B | Congrats, <name>! You won $<Points> | Congrats, <name>! You won $<Points> | Passed |

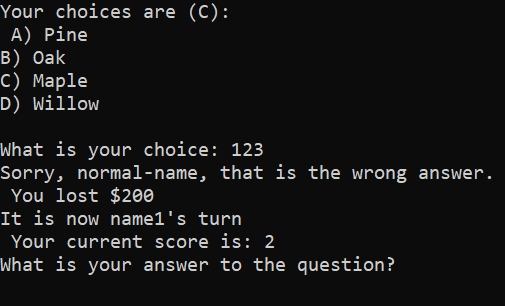


A screenshot of a computer screen

Description automatically generated

**Wrong Answer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A wrong answer is given to pass the turn to the next players | 123 | Sorry, <name>, that is the wrong answer. You lost <$Points>  It is now <next player>’s turn | Sorry, <name>, that is the wrong answer. You lost <$Points>  It is now <next player>’s turn | Passed |



Function: void ChangePlayers(int \*nTurnTracker, int nPlayers,

                   char\* strFirst, char\* strSecond, char\* strThird,

                   int\* nFirstScore, int\* nSecondScore, int\* nThirdScore,

                   char\* strActivePlayer, int\* nActiveScore)

Function Description: Based on the nTurnTracker it applies the changes to the active player. The nTurnTracker changes and sets the strActivePlayer and nActiveScore to the next player’s details. nTurnTracker only goes as high as the nPlayers then resets it back to 0 to never overflow.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A wrong answer is given to pass the turn to the next players. Goes to player 3. | zzz | Sorry, <name>, that is the wrong answer. You lost <$Points>  It is now <next player>’s turn | Sorry, <name>, that is the wrong answer. You lost <$Points>  It is now <next player>’s turn | Passed |

A computer screen shot of a black background

Description automatically generated

A screenshot of a computer screen

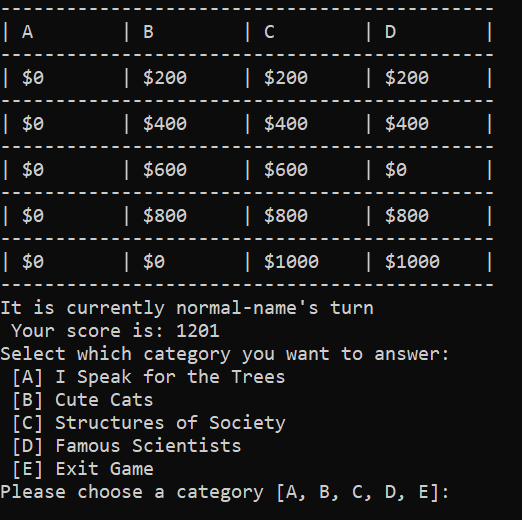
Description automatically generated

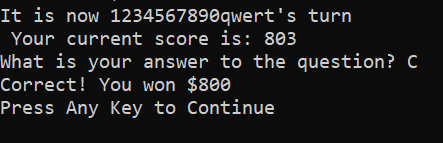
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| A wrong answer is given 3 times in a row. Returns to the initial player. | b  b  b | Nobody got the correct answer.  It is now <initial player>’s turn | Nobody got the correct answer.  It is now <initial player>’s turn | Passed |

A screenshot of a computer

Description automatically generated

**As written in the length in the beginning, the Jeopardy round will only go for 8 turns.**

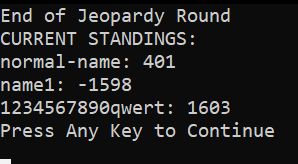
** 7 rounds finished**

****

**-End of PickChoice being the topic function-**

**Topic Function is PlayGame again**

After the nJeopardyProgress becomes 0, Jeopardy round ends. Current standings are shown and the Double Round starts. For the sake of time, the same questions from question1 are reused but have z at the start



**Double Round print screenshot is from a different session which causes mismatch.**

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

Function: void PlayGame2(int\* nLength2, int nPlayersHold, int nTurnTrackerHold,

char\* strFirstHold, char\* strSecondHold, char\* strThirdHold, char\* strActivePlayerHold,

int nFirstScoreHold, int nSecondScoreHold, int nThirdScoreHold, int nActiveScoreHold, int nDoubleMode,

int\* nFinal, int\* nFinaleScore1, int\* nFinaleScore2, int\* nFinaleScore3)

Function Description: Similar to PlayGame() but accesses nFinal and nFinaleScore# variables which will be updated at the end of the round. It does not contain the initializing parts such as getting the number of players and player names. GetContent commands are the same but instead takes content from questions2.c.

In the Double Round, the point values are doubled and there are 2 penalty modes based on nDoubleMode. Normal and Zero.

In normal mode, mistakes just deduct from your points

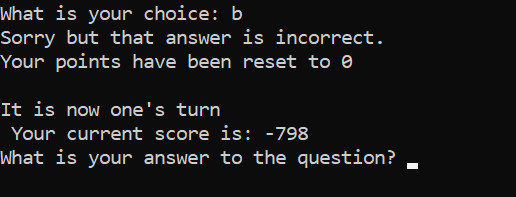
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Normal mode mistake | A | Sorry, <player name>, that is the wrong answer.  You lost $<Points>.  It is now <next player>’s turn | Sorry, <player name>, that is the wrong answer.  You lost $<Points>.  It is now <next player>’s turn | Passed |

A black screen with white text

Description automatically generated

In Zero mode, a mistake makes your score 0 regardless of whether it was positive or negative.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| Zero mode mistake | b | Sorry, <player name>, that is incorrect.  Your points have been reset to 0.  It is now <next player>’s turn | Sorry, <player name>, that is incorrect.  Your points have been reset to 0.  It is now <next player>’s turn | Passed |

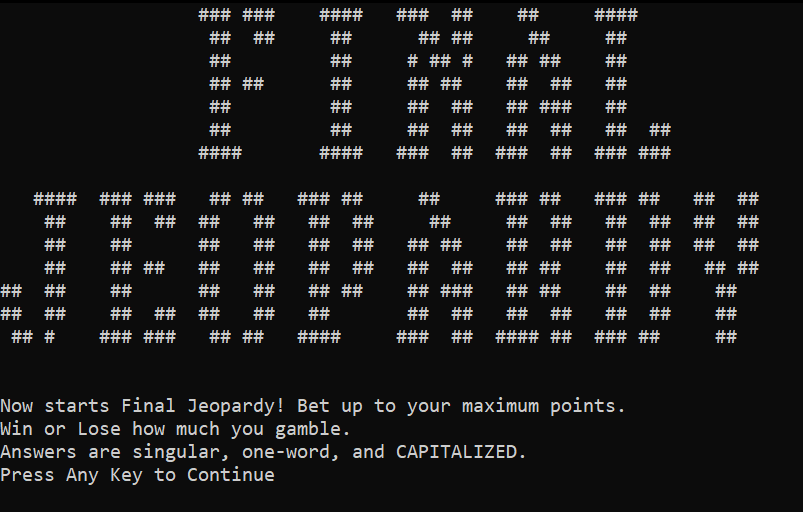


A screenshot of a computer screen

Description automatically generated

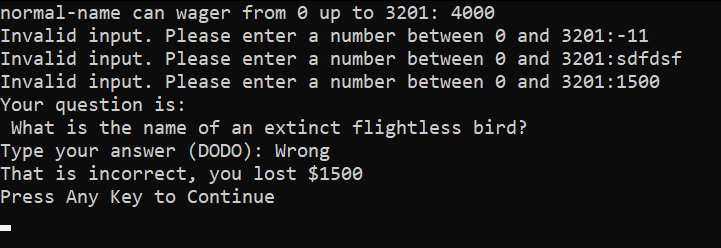
At the end of the Double round, it will reveal the score standings and lead to the Final Jeopardy.

**Topic Function is main() again**

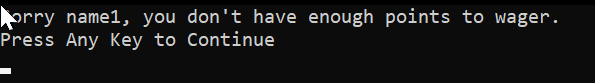
****

**You may wager if your points are greater than 0 to answer a random questiion. If you have 0 or less, you wont’t be able to wager.**

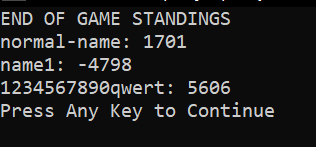
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Description** | **Inputs** | **Expected Outputs** | **Actual Output** | **P/F** |
| An invalid input. | -11 | Invalid input. Please enter a number between 0 and %end if double round score | Invalid input. Please enter a number between 0 and %end if double round score | Passed |
| An invalid input. | sdfdsf | Invalid input. Please enter a number between 0 and %end if double round score | Invalid input. Please enter a number between 0 and %end if double round score | Passed |
| A valid input | 1500 | An valid input is added | An valid input is added | Passed |

****

**If not enough points:**

****

**At the end show ending game standings**

****

**Side details**

The usage of one or two players only works just fine everywhere else. The game functions by modifying nActiveplayer and nActiveScore which only get set dependinding on the state of nTurnTracker which won’t reach values past the nPlayers.

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generatedUpon finishing a game, you are returned to the main menu to play again since the functions are within a while(1) where the only way to exit is selecting 3.

