OCR computer science J276

March 2020

Name: Harry Carr

Candidate Number:

St Aidan’s C of E High School

Centre Number: 48237

Contents

[1.0 Analysis 3](#_Toc35004423)

[1.1 Brief 3](#_Toc35004424)

[2.0 Design 4](#_Toc35004425)

[3.0 Test Plans 5](#_Toc35004426)

[4.0 Implementation 6](#_Toc35004427)

[5.0 Testing 7](#_Toc35004428)

[6.0 Evaluation 8](#_Toc35004429)

# Analysis

# Brief

I have been asked to develop a dice game that simulates two-player’s rolling dice. There are 5 rounds and each round two dice rolls are rolled. Each roll consists of two dice and the two numbers are added together to give their total score. This total is then added to their score, however, there are addition ways to lose and gain score depending on what they role. At the end of the game the person with the highest score will win. The score and name are then recorded and the top 5 scores are displayed.

# Objectives

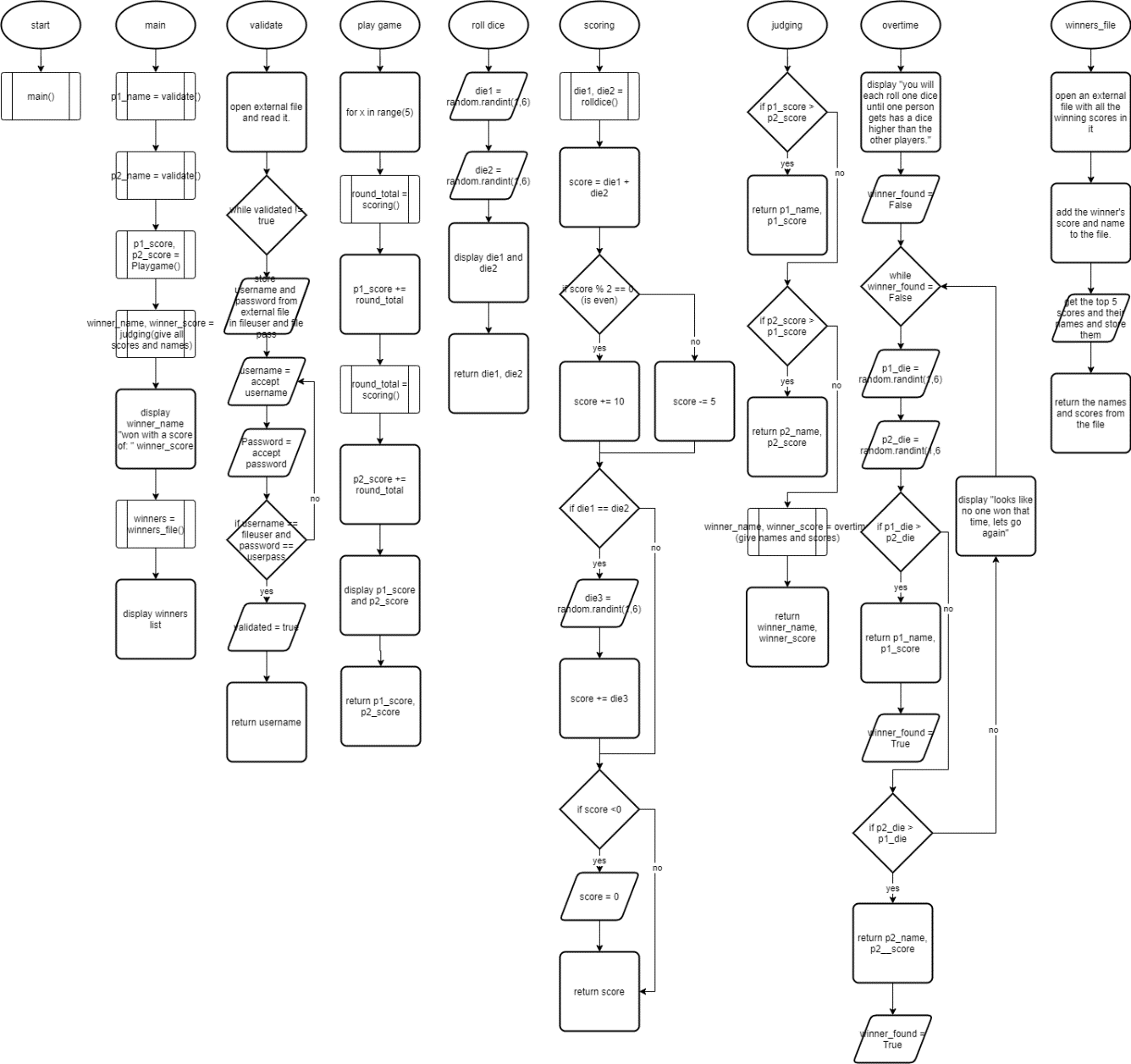
The program should:

* + - It should display a welcome message
    - Allow two players to enter their details to be authenticated to ensure that they are authorised players. The program must read data from an external file to authenticate the user
    - Allow each player to roll two 6-sided dice and store the rolls.
    - Calculates and outputs the points for each round and each player’s total score
    - Allow the players to play 5 rounds
    - If both players have the same score after 5 rounds it should allow each player to roll 1 die each until someone wins.
    - Outputs who has won after 5 rounds have been played
    - Stores the winners name and score in an external file
    - Displays the score and player name of the top 5 winning scores from an external file
    - It should validate inputs from users where

Possible

# Design

* 1. algorithm Plan
     + accept a username and password from player 1
* validate it against details from an external file
  + - accept a username and password from player 2
* validate it against details from an external file
* start a 5-round loop
  + - allow player 1 to roll two dices
* find the total of the two dices and add it to the score
* check if the total of the is even
* if so add another 10 points to score
* check if the total is odd
* if so subtract 5 from score
* check if they user has rolled a double
* if so allow one extra roll and add the total of the roll to their score
* check if score is less than 0
* if so score = 0
  + - repeat the roll and scoring process for player 2
    - after 5 rounds end the loop
    - check if player1’s score is higher than player2’s
* if player 1 win’s display “player 1 won”
* else if player2’s score is higher than player1’s display “player 2 won”
* else display “draw, you each get 1 dice and whoever gets the highest score wins”
* start a while loop until one is bigger than the other
* roll a dice for player 1
* roll a dice for player 2
* if player 1’s roll is larger than player2’s display name “wins” and end loop
* else if player2’s is larger than player1’s then display name “wins” and end loop
* else carry on loop.
* Store the winners score and name in an external file
* Display the name and score of the top 5 scores.
  1. Flowchart



* 1. Data structures

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Example Data | Validation |
| username | string | “Harry” | Matches a valid user name |
| password | string | “4321” | Matches a valid user password |
| P1\_name | string | “Kareem” |  |
| P2\_name | string | “Amogh” |  |
| Die1 | integer | 5 |  |
| Die2 | integer | 1 |  |
| Die3 | integer | 6 |  |
| validated | Boolean | true |  |
| P1\_score | integer | 20 | Not less than 0 |
| P2\_score | integer | 5 |  |
| Winner\_name | string | “Nikhil” |  |
| Winner\_score | integer | 50 |  |
| score | integer | 4 |  |
| Round\_total | integer | 2 |  |
| File\_pass | string | “4321” |  |
| File\_user | string | “George” |  |

* 1. Pseudocode

# Test Plans

* 1. objective test plan

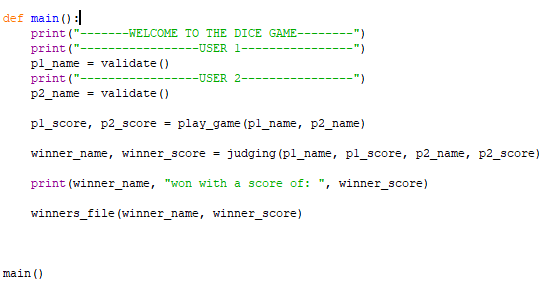
|  |  |  |  |
| --- | --- | --- | --- |
| Objective | What should happen | What happened | Improvement |
| Logging in | user gives a username and password and it should accept them or reject them after checking them |  |  |
| Signing up | User gives a desired username and password and the program should write these to the external file where the usernames and passwords are kept and then loop back to allow them to log in again |  |  |
| scoring | The program should get two random dice rolls and do the necessary actions to them like checking that the sum is equal or odd and whether the dice are equal and add the correct amount to the players score. This should be repeated for each player. The score should never go below 0. |  |  |
| Output scores | It should output the name and the score that each player is currently on. |  |  |
| overtime | If the score is a tie the program should automatically go into overtime to find the winner |  |  |
| leader board | At the end of the game the program should add the winning score to an external file with all the scores on and then get the top 5 scores and display them. |  |  |

* 1. data entry test plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testing What | Data | What should happen | What happened | improvements |
| Enter username | Harry | Accept |  |  |
| harry | reject |  |  |
| 1234 | reject |  |  |
| Enter Password | 9 | Accept |  |  |
| 10 | reject |  |  |
| abcd | reject |  |  |

# Implementation

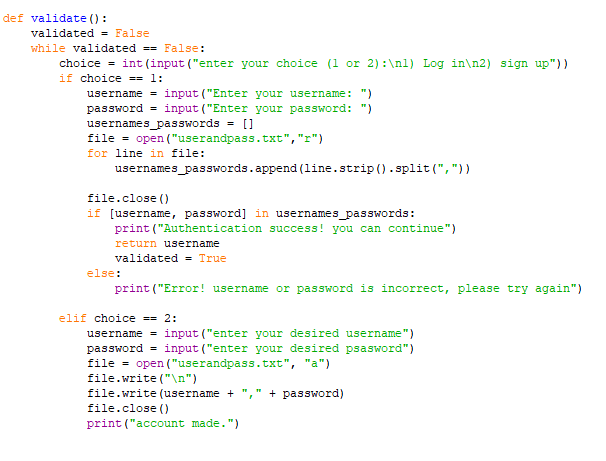
main function that controls the game:

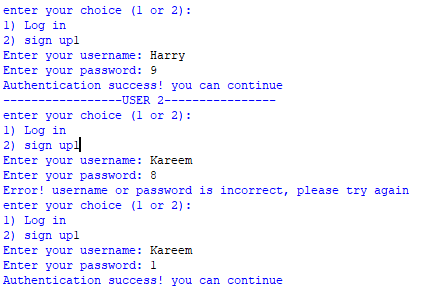


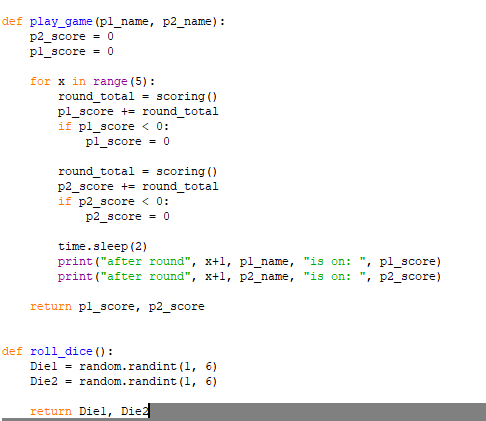


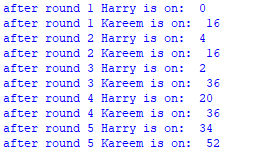




The validation function that controls logging and signing in: 



The play game function that runs and controls the game: 



# 5.0 Testing

# 6.0 Evaluation