ZC – University of Science and Technology

Communications & Information Engineering Program

CSCI 101: Introduction to Computer Science

Dominos Game

Team: A&A

Team Members:

202300938 Abdullah Ibrahim Afifi [s-abdullah.afifi@zewailcity.edu.eg](mailto:s-abdullah.afifi@zewailcity.edu.eg)

202301016 Amr Emad Mohamed [s-amr.mohamed@zewailcity.edu.eg](mailto:s-amr.mohamed@zewailcity.edu.eg)

Team contact:

[s-abdullah.afifi@zewailcity.edu.eg](mailto:s-abdullah.afifi@zewailcity.edu.eg)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project description:

Initialization:

scores = [0]: Initializes a list with a single element, 0. This list will store the scores of individual players.

w = 0: Initializes the variable w to 0, representing the initial value for the player index in the start\_game function.

N = get\_correct\_N(): Calls the get\_correct\_N function to obtain the number of players (N).

player, data = Efile(N): Calls the Efile function to get information about playing players in the player dictionary and the database in the data dictionary from the file 'update\_scores'.

Main Game Loop:

The main game loop runs as long as the maximum score among players is less than or equal to 10, indicating that a new game should start.

Inside the loop, a new game is initiated, and players are provided with a set of domino cards.

The start\_game function is called to initialize the game, determining the starting player and the initial ground setup.

Players take turns playing domino cards until one player's score exceeds 10.

Player Turns:

Each player takes turns playing domino cards based on specific rules.

Players can choose cards to play, withdraw if they cannot play, or pass if they don't have suitable cards.

The game continues until all players have played their cards or one player wins the round.

Scoring and End of Game:

After each round, the scores are updated based on the winner of the round.

The loop continues until a player's score exceeds 100 points.

The champion(s) with the highest score are identified, and their data is updated in the database (data).

The updated database is then written back to the file 'update\_scores' using the Update\_file function.

Print Information:

The code prints information about the players, including their scores and the game's status.

In summary, this code simulates multiple rounds of a text-based Domino game, tracks player scores, and updates a database with game statistics. It provides a simple yet interactive experience for playing the game.

The updated database is then written back to the file 'update\_scores' using the Update\_file function.

Print Information:

The code prints information about the players, including their scores and the game's status.

In summary, this code simulates multiple rounds of a text-based Domino game, tracks player scores, and updates a database with game statistics. It provides a simple yet interactive experience for playing the game.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function name | Input parameters | Output of the function | Description | Team member |
| Generate\_ Domeno\_ cards | \_ | List of domino cards | Generating domino cards and shuffling it. | Abdullah |
| Get\_ correct\_ N | \_ | Number of players | Ask user about the number of players and printing Error message if he Entered an invalid number | Amr |
| Distribute­\_ cards | Cards \ number of players | Advanced list of players’ cards | it Uses the output of the fun Generate \_Domeno \_cards and appends 7 cards to each player and delete them from card list | Abdullah |
| Start\_ game | number of players | the Index of the starting player | append the first card to the ground | It searches for the first card in each player using while loop, append it to the ground and remove it from the player | Abdullah |
| next­\_ player | number of players \ player index | Player index | It adds 1 to the index after each turn and put it zero when it’s = number of players | Amr |
| update\_ ground | ground string \ played card | ground string | It stores the value of the first and last card and looks at the edge of the played card and compare them if the player could play in both sides, it asks him which side | Amr |
| calc\_ weights | number of players \ player’s cards | Players\_weights | It calculates the total points of each player and save it in a list. | Amr |
| E\_file | number of players | \_ | It reads player’s ids and check if is it in the file and writes new players’ data | Abdullah |
| End\_game | Players’ (cards/weights/ scores)/ground cards | Players’ new scores | It checks if the player had finished his cards, or all the players don’t have cards to play then determine the winner and calculate his score | Amr |
| Update\_File | Player’s scores | \_ | Update the new score for each player in the file | Abdullah |

A diagram of a game

Description automatically generated