Assignment 1 – Pass the Pigs

Aiden Trager

CSE 13S – Fall 2023

Purpose

This is a program to simulate the game Pass the Pigs. This is an entirely luck based game focused around the idea of rolling a five sided pig. Not all sides are equally likely to occur and each side has a point value. You will accumulate points over time and the game will end when one person gets over 100 points. This will exemplify basic C programming and be a good jumping off point for future projects.

How to Use the Program

In order to run the program first make the program and then run it. These are what the commands should look like in your terminal:

```
make pig
```

Followed by:

```
./pig
```

The program starts by prompting the user to input the number of players (between 2 and 10 inclusive). An error will be given if the value is not in the range. You will then be prompted for a random number seed. The game will then run until a player wins.

Program Design

Program starts by defining and setting variables.
Variables:

- `num_players`: Number of players.
- `seed`: Random number seed.
- `win`: Flag indicating if a player has won.
- `turn`: Flag indicating if it's the player's turn.
- `roll`: The result of the dice roll.
- `i`: Loop variable.
- `scores`: Array storing the scores of each player.
- `amount_gained`: The score gained by a player in a turn.

These are the different variables present in the code and what they represent. There is one main algorithm and function described below.

Data Structures

Structures:

Enum for Pig Positions: Defines the possible positions a pig can have (SIDE, RAZORBACK, TROTTER, SNOUTER, JOWLER).

pig: Array representing pig positions based on pseudorandom numbers.

player_name: Array of player names.

These are arrays and Enum that were given in the assignment description. They help form the basics of the game. I chose them because they were given.

Algorithms

Algorithms:

- The program uses a nested loop structure to simulate turns for each player until one player reaches a score of 100.
- Inside the inner loop, a dice roll determines the pig's position, and the player's score is updated accordingly.
- There are conditional statements to handle different pig positions, updating scores and checking for a winning condition.

```
// Main game loop
while win is not true:
    // Loop through each player's turn
    for each player in players:
        print(player)

        // Player's turn loop
        while is player's turn:
            roll = getRandomNumber() % 7
            // Evaluate pig position and update score
            // Check for win condition
            if player has won:
                win = true
```

Function Descriptions

My program will only have one function made by me, the main function.

Inputs: None

Outputs: Returns 0 (integer) to indicate successful execution.

Purpose:

- The main function serves as the entry point of the program.
- It collects user inputs for the number of players and the random seed.
- It initializes the game variables, sets the random seed, and initializes the scores array for each player.
- It contains the main game loop, where players take turns rolling a virtual pig-shaped dice until one player reaches a score of 100.

```
Collect and validate user inputs for num_players and seed
Initialize game variables and set the random seed
Initialize scores array for each player

while the game is not won:
    for each player:
        print player’s name
        while it’s the player’s turn:
            roll the virtual pig-shaped dice
            update player’s score based on the pig’s position
            if the player’s score is >= 100:
                end the game
```

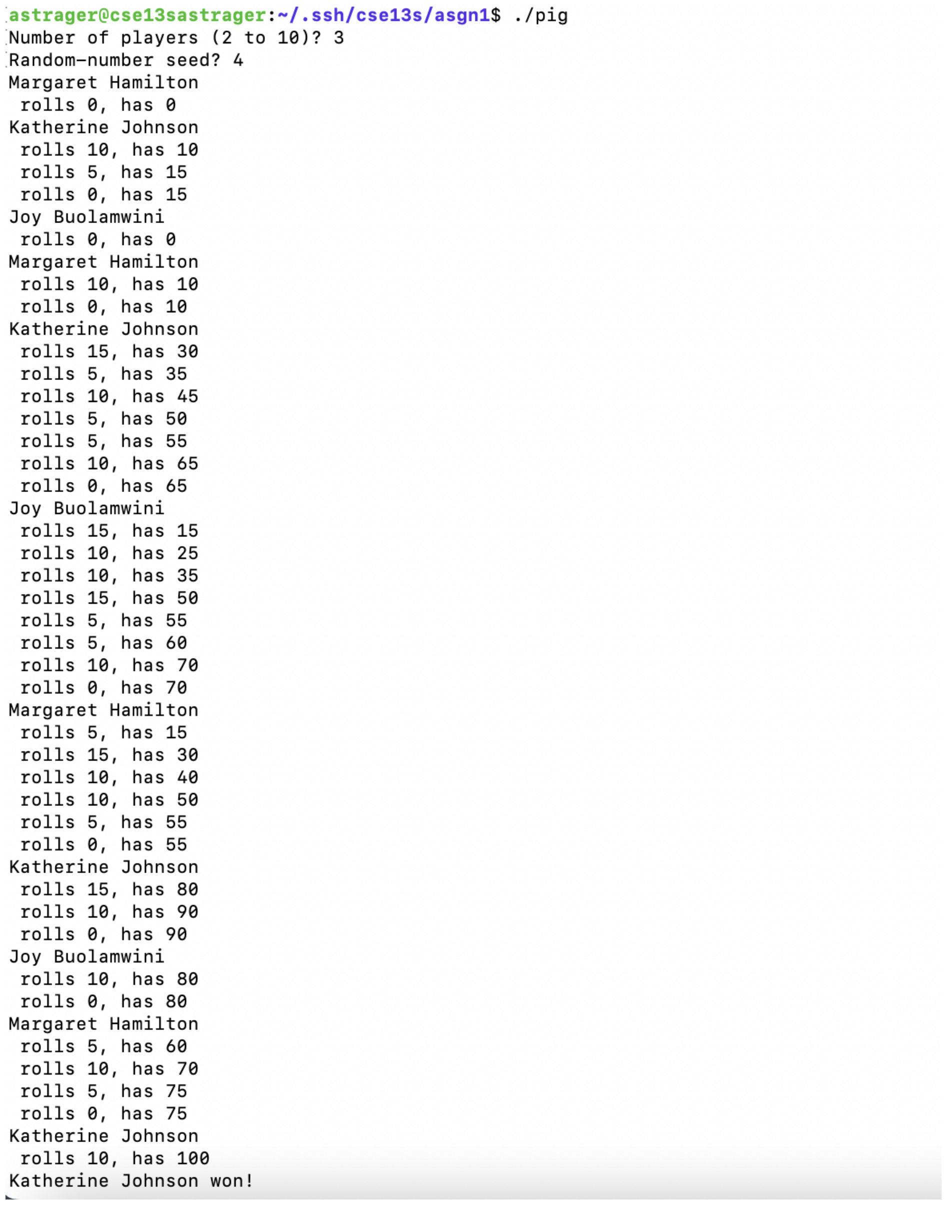
Decision Making:

- The use of a while loop ensures the game continues until a winner is determined.
- Nested loops handle player turns and dice rolls, providing a clear structure for gameplay.

Results

In completing this assignment, I’ve designed a simple text-based game in C that simulates a turn-based pig dice game for a variable number of players. The program collects user inputs for the number of players and the random seed, initializes game variables, and uses a combination of loops and switch statements to manage gameplay. It works as intended. My output is below for a specific input.

I did not use anything as a Reference.



```
astrager@cse13sastrager:~/.ssh/cse13s/asgn1$ ./pig
Number of players (2 to 10)? 3
Random-number seed? 4
Margaret Hamilton
 rolls 0, has 0
Katherine Johnson
 rolls 10, has 10
 rolls 5, has 15
 rolls 0, has 15
Joy Buolamwini
 rolls 0, has 0
Margaret Hamilton
 rolls 10, has 10
 rolls 0, has 10
Katherine Johnson
 rolls 15, has 30
 rolls 5, has 35
 rolls 10, has 45
 rolls 5, has 50
 rolls 5, has 55
 rolls 10, has 65
 rolls 0, has 65
Joy Buolamwini
 rolls 15, has 15
 rolls 10, has 25
 rolls 10, has 35
 rolls 15, has 50
 rolls 5, has 55
 rolls 5, has 60
 rolls 10, has 70
 rolls 0, has 70
Margaret Hamilton
 rolls 5, has 15
 rolls 15, has 30
 rolls 10, has 40
 rolls 10, has 50
 rolls 5, has 55
 rolls 0, has 55
Katherine Johnson
 rolls 15, has 80
 rolls 10, has 90
 rolls 0, has 90
Joy Buolamwini
 rolls 10, has 80
 rolls 0, has 80
Margaret Hamilton
 rolls 5, has 60
 rolls 10, has 70
 rolls 5, has 75
 rolls 0, has 75
Katherine Johnson
 rolls 10, has 100
Katherine Johnson won!
```