Assignment 1 – Pass the Pigs

Aditya Bhaskar

CSE 13S – Spring 2023

Purpose

The plan is to create a modified version of the dice game 'Pass the Pigs,' invented by David Moffat. Our objective is to design a simplified version of the game, suitable for learning introductory programming in the C language. This game does not require any skills or strategic decision-making, making it a perfect fit for beginners who are just starting to learn programming.

The game development will involve utilizing all the fundamental programming language facilities, including primitive data types, loops, conditionals, arrays, and handling user input. These programming constructs are essential for developing any software application, and mastering them is critical for becoming a proficient programmer. Therefore, implementing Pass the Pigs will enable us to understand these concepts better and put them into practice.

In summary, our aim is to create a fun and engaging game that is both easy to play and teaches programming fundamentals. By working on this project, we will gain practical experience in programming while developing their problem-solving and critical thinking skills.

How to Use the Program

Our goal is to develop a simplified version of Pass the Pigs that can be played by a group of k players, where k is between 2 and 10. The game involves players taking turns rolling an asymmetrical die, affectionately called the pig, to earn points. Each roll can result in the pig landing in one of five different positions, each with a corresponding probability, point value, and next action. The positions are Jowler, Razorback, Trotter, Snouter, and Side, with probabilities of 2/7, 1/7, 1/7, 1/7, and 2/7, respectively. Jowler is worth 5 points, Razorback and Trotter are worth 10 points each, Snouter is worth 15 points, and Side results in zero points and ends the player's turn.

The game continues until one of the players scores 100 points or more, at which point the game ends. The rules for the next action following each position are as follows: if the pig lands on Jowler, Razorback, Trotter, or Snouter, the player rolls again, while landing on Side ends the player's turn, and the next player rolls.

To play the game, the user must input the number of players and a seed value to initialize the random number generator. The program will then simulate the game, keeping track of each player's score and displaying the results at the end of each turn.

In summary, our program allows a group of players to enjoy a fun and simple game that helps us practice our programming skills. By implementing Pass the Pigs, programmers can learn about probability, random number generation, and data structures such as arrays. Furthermore, the game's simple rules and mechanics make it accessible to a wide range of players, regardless of their skill level or programming experience.

Program Design

First, we add some important header files such as: **stdio.h** (for standard input/output), **stdlib.h** (for the implementation of *rand()* and *srand()* functions), and **time.h** (this header file is not necessary but I added it anyway just to be sure). Then we define some constants at the top of the program using the *#define* preprocessor directive. *GOAL_SCORE* sets the target score that a player needs to reach to win the game. Lastly, we include the **names.h** header file for all the names.

Next, we define an enumeration type called *pig_position* to represent the five possible positions that the pig can land on when rolled. These positions are **JOWLER**, **RAZORBACK**, **TROTTER**, **SNOUTER**, and **SIDE**. We also define the constant array *pig* of **pig_position** datatype. This is done in order to add weight to the sides. In this case, the probability of the side **SIDE** is $\frac{2}{7}$, probability of the side **SNOUTER** is $\frac{1}{7}$, probability of the side **TROTTER** is $\frac{1}{7}$ probability of the side **RAZORBACK** is $\frac{1}{7}$, and probability of the side **JOWLER** is $\frac{2}{7}$.

We then define two functions, **roll_pig()** and **play_game()**. **roll_pig()** generates a random number between 0 and 6 (inclusive) and returns an integer value corresponding to one of the five pig positions. This function will be used to simulate rolling the pig during the game.

play_game() is the main function that runs the Pass the Pigs game. It takes two arguments: the number of players and a seed value used to initialize the random number generator. Within the function, we initialize an array scores to keep track of each player's score, an integer **current_player** to keep track of whose turn it is, an integer **winner** to keep track of who wins, integers **jval, rval, tval, sval, val** to store the integer values of every side, and an integer **turn** to check whether the turn is over or not.

We then use the **srand()** function to seed the random number generator based on the seed argument passed to **play_game()**.

We then enter a loop that continues until a player reaches the **GOAL_SCORE** and wins the game. Within the loop, we use **roll_pig()** to simulate rolling the pig and determine the pig's position. We then enter the second while-loop to make sure the player keeps rolling until their turn ends. We print out which player rolled the pig and what position it landed on using a switch statement. If the pig lands on JOWLER, RAZORBACK, TROTTER, or SNOUTER, we add the corresponding number of points to the current player's score and print out their updated score. If the pig lands on the SIDE, the current player's turn ends, and we move on to the next player.

After each turn, we check if the current player's score is equal to or greater than **GOAL_SCORE**. If it is, we declare the current player as the winner, and the game ends.

Finally, we print out the winner's name.

The main function starts by prompting the user to enter the number of players and a seed value for the random number generator. We then check if the number of players / seed is within the allowed range and return an error message and use the default value of 2 / 2023 if it isn't.

Data Structures

An integer array called **scores**, which stores the current score for each player. The array is initialized to zero at the beginning of the **play_game** function and is updated after each roll of the pig. We also define **pig** which is an *enum* data structure. This data structure is used to make the pig-dice asymmetrical. Another array **player_names** which is a string array is used to give names to players and is included in the **names.h** header file.

Algorithms

```
while (winner == -1){
    turn = 1
    while (turn == 1){
        position = roll_the_pig();
        switch(position){
            case 1:
                scores += points;
                break;
            .
            .
            .
            case 5:
                scores += points;
                current_player = (current_player + 1) % num_players;
                turn = 0;
                break;
        }
        if score[current_player] >= 100 {
            winner_score = score[current_player];
            winner = current_player;
            break;
        }
    }
}
```

Function Descriptions

1. **int roll_pig(void):**
   - This function rolls a die and returns the position of the pig.
   - It does not take any arguments.
   - It returns an integer value representing the position of the pig
2. **void play_game(int num_players, unsigned int seed):**
   - This function plays the Pass the Pigs game.
   - It takes two arguments:
     - **num_players**: an integer value representing the number of players in the game.
     - **seed**: an unsigned integer value used to seed the random number generator.
   - The function does not return anything.
   - It initializes the game with zero scores for all players.
   - It takes turns for each player, rolling the pig and updating the scores accordingly.
   - It checks for a winner at the end of each turn and ends the game if a winner is found.
   - It prints the winner and their score at the end of the game.

Results

This code is a basic implementation of the Pig Dice Game where players take turns rolling a pig dice and accumulating points until one player reaches a pre-defined goal score. The code includes the implementation of the Pig dice, probabilities for each pig position, and logic for determining the winner.

The program prompts the user for the number of players and a random number seed. If the user input is invalid, the program defaults to 2 players and a seed value of 2023.

The code consists of a single file with separate functions for rolling the pig dice and playing the game. The code is easy to read and understand due to proper use of naming conventions and comments.

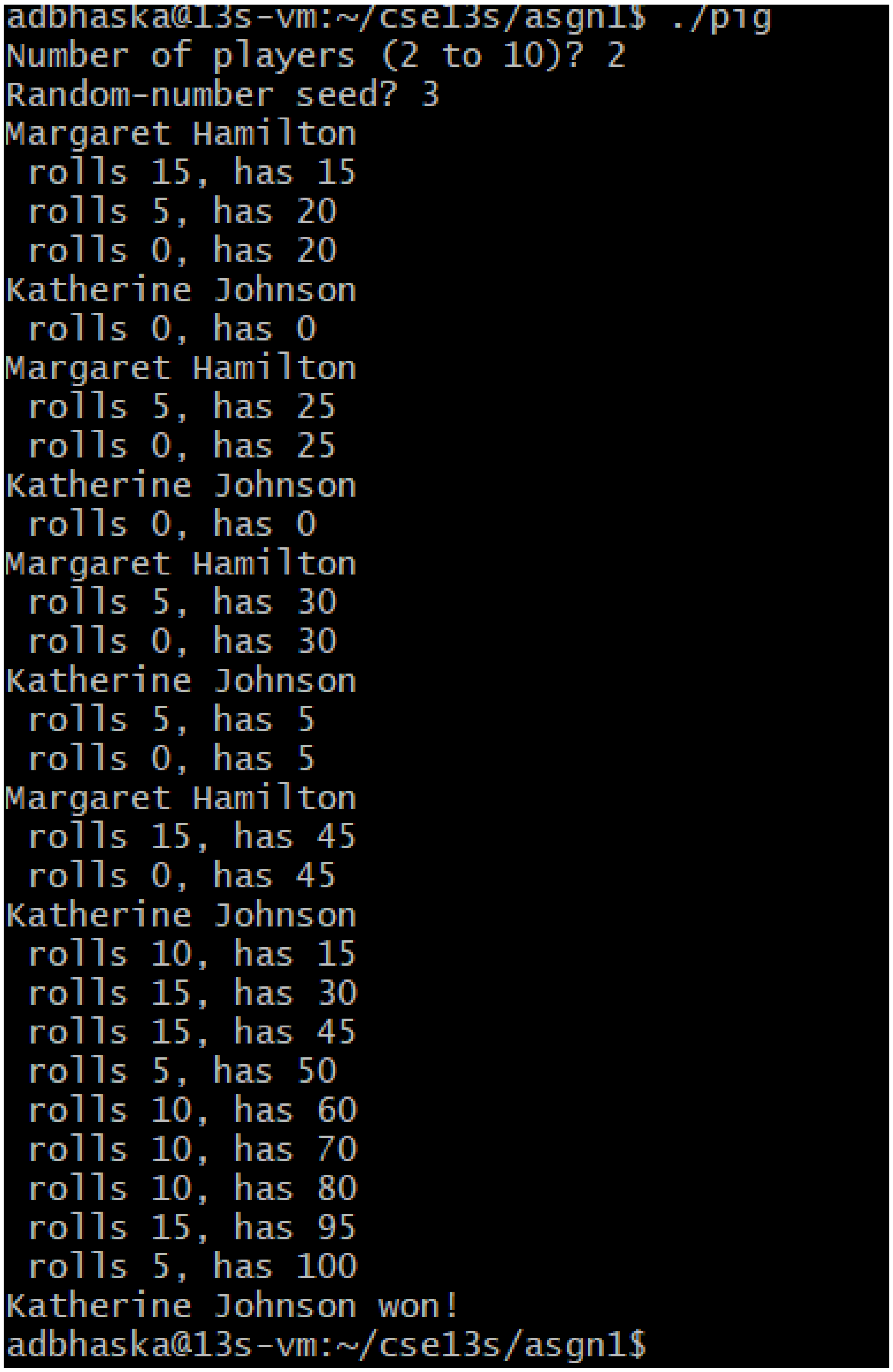
The program correctly calculates and displays each player's score after each turn. Once a player reaches the goal score, the program correctly identifies the winner and terminates the game.

Overall, the code successfully achieves everything it is intended to do. However, there are some areas for improvement. The program could benefit from more user-friendly prompts and error messages when invalid input is entered. Additionally, the code could be expanded to include more game features such as allowing players to choose their names or selecting different goal scores.

In summary, the code satisfies the requirements of the assignment and is well-organized and easy to read. With some minor improvements, it could be further enhanced to provide a better user experience.

Here is a screenshot of the program Fig. 1.

References



```
adbhaska@13s-vm:~/cse13s/asgn1$ ./pig
Number of players (2 to 10)? 2
Random-number seed? 3
Margaret Hamilton
 rolls 15, has 15
 rolls 5, has 20
 rolls 0, has 20
Katherine Johnson
 rolls 0, has 0
Margaret Hamilton
 rolls 5, has 25
 rolls 0, has 25
Katherine Johnson
 rolls 0, has 0
Margaret Hamilton
 rolls 5, has 30
 rolls 0, has 30
Katherine Johnson
 rolls 5, has 5
 rolls 0, has 5
Margaret Hamilton
 rolls 15, has 45
 rolls 0, has 45
Katherine Johnson
 rolls 10, has 15
 rolls 15, has 30
 rolls 15, has 45
 rolls 5, has 50
 rolls 10, has 60
 rolls 10, has 70
 rolls 10, has 80
 rolls 15, has 95
 rolls 5, has 100
Katherine Johnson won!
adbhaska@13s-vm:~/cse13s/asgn1$
```

Figure 1: Screenshot of the program running.