Pass The Pigs - CSE13S Assignment 1

Nicholas Reis

September 30, 2021

Game Setup

This program will simulate a game involving rolling pigs and gaining points based off of the orientation in which they land. The program will take two inputs from the user before it begins:

1. The number of players that will play the game, ranging between 2 and 10 people. If the user inputs an invalid number of players, the following error will be printed to stderr: "Invalid number of players. Using 2 instead"
2. A random seed for the next game to be played. This allows for tests to be replicated through the generation of psuedorandom numbers that remain the same if the same seed is used. If the user inputs an invalid random seed, the following error will be printed to stderr: "Invalid random seed. Using 2021 instead."

Game Rules

Once this information has been correctly input, the program will begin the game starting from player 0. Each player begins the game with 0 points, and they can gain points depending on which way the pigs land. Rolling Side (2/7 chance) will give the player 0 points and then end their turn and move to the next player, rolling Jowler (2/7 chance) will give the player 5 points, rolling Razorback or Trotter (1/7 chance each) will give the player 10 points, and rolling Snouter (1/7 chance) will give the player 15 points. The first player to reach 100 points in the game will win.

Pseudocode

```
include names.h
include limits.h
include stdio.h
include stdlib.h
include stdint.h
```

```
define SEED 2021

create an enum with each position for the pig (7 possibilities)

main:

define an int variable playerCount
ask the user to input a player count between 2 and 10
set playerCount to user input
if playerCount is less than 2 or greater than 10:
    print an error message
    set playerCount to 2

define a long variable seed
define a long variable max_uint
ask the user to input a random seed
set seed to the user input
if seed is less than 0 or greater than max_uint:
    print an error message
    set seed equal to SEED
use srandom() with seed as the argument for replicability

create an array to hold player scores (length = number of players)

for loop iterates through each array element
    initialize every score to 0

create a variable for current player
for loop iterates through each player's turn until one player reaches 100 points
    int roll equals the value of random() % 7

    switch (roll)

    case 0 and 1:
        roll side and get 0 points added
        if the current player is the last player
            go back to player 0
        turn ends and moves on to next player
    case 2:
        roll razorback and get 10 points added to current player's score
        break to beginning of loop
    case 3:
        roll trotter and get 10 points added to current player's score
        break to beginning of loop
    case 4:
        roll snouter and get 15 points added to current player's score
```

```
        break to beginning of loop
    case 5 and 6:
        roll jowler and get 5 points added to current player's score
        break to beginning of loop
    default:
        here for any other cases (although impossible)
        break to beginning of loop

if current player's score equals 100
print congratulatory message

End Program
```

Pseudocode Explained

- includes and defines

  The libraries are used primarily for printing to stderr and stdout along with the srandom() and random() functions. The macro is used in case the user enters an invalid seed.

- section that prompts the user for the player count

  This portion of code simply takes an integer input from the user and sets it to the "player_count" variable. If they input a number above 10 or below 2, the program sets player_count to 2.

- section that prompts the user for the seed

  This portion of code takes in an unsigned integer from the user and sets it to the "seed" variable. If they input a number below zero or above uint_max, the program sets the seed to 2021.

- use srandom() with seed as the argument for replicability

  srandom() is used so that the program utilizes pseudo-random numbers and can thus be replicated by graders.

- int roll equals the value of random() % 7

  I set the int variable "roll" to be equal to the outcome of random() % 7, which takes the remainder of the random number divided by 7. This effectively works as a dice roll, with the number being somewhere between 0 and 6, as there are 7 ways the pig can be rolled.

- switch and case section

  Depending on what value "roll" is, the switch will go to a specific case (between 0 and 6). If the roll is either 0 or 1, the player rolls SIDE, meaning they get no points and their turn is over. If they roll a 2, they roll RAZORBACK, meaning they get 10 points and get to

roll again. If they roll a 3, they roll TROTTER, meaning they get 10 points and get to roll again. If they roll a 4, they roll SNOUTER, meaning they get 15 points and get to roll again. If they roll a 5 or a 6, they roll JOWLER, meaning they get 5 points and get to roll again.

Design Process

- Began writing pig.c by adding the #include C libraries
- Added main() and the enumeration code block from the assignment pdf
- Implemented the code that prompts the user for the player count (only between 2 and 10)
- Implemented the code that prompts the user for a seed (greater than 0 and less than uint_max)
- Used the seed as an argument in srandom()
- Created an array with the length of playerCount to keep track of player scores
- Used a for loop to initialize all the scores to 0
- Created a for loop that iterates through each player until one reaches 100 points
- Used the random() function and %7 to randomize the rolls
- Utilized a switch to cover cases 0 through 6 (each possible way the pig lands)
- Each roll aside from SIDE added points to the current player, SIDE ended a player's turn
- Realized the increment was messing up my code, had to include a decrement to allow a player to continue their turn
- Added ending print statement to stdout with the name of the winner
- Looked through the style guide and edited some of the code to fit the requirements
- Used the "diff" command to compare my code output with the file in resources
- Made changes to my code to make it match the output
- Made changes to the section of code with seed information as the boundaries were not working