Assignment 1 - Pass the Pigs

Teresa Joseph

CSE 13S - Professor Long

Fall 2021 - September 30

Purpose

Implement a game of "Pass the Pigs" where players take turns rolling a pig to score points.

Game Breakdown

- k number of players (2 min and 10 max)
- Each player rolls a pig of 5 possible positions to gain points
    - 0 points for side (2/7 probability)
    - 5 points for jowler/ear (2/7 probability)
    - 10 points for razorback/back (1/7 probability)
    - 10 points for trotter/upright (1/7 probability)
    - 15 points for snouter/nose (1/7 probability)
- Continue rolling until side is rolled (then pass to next player) or reach 100 or more points (then win and end the game)

Pseudocode

Libraries to include

<stdio.h> for general commands

<stdlib.h> for random generator

<limits.h> for seed range

"names.h" for accessing player names

Main function:

[Obtaining the number of players]

Prompt user to enter the number of players and label the input as *k*

If *k* is less than 2 or greater than 10:

Print error message and assume 2 players (so *k* = 2)

If *k* is not an integer:

Print error message and assume 2 players (so *k* = 2)

Else, continue with *k* = user's input

[Obtaining the seed value]

Prompt user to enter a valid unsigned seed value and label the input as *seedValue*

If *seedValue* is greater than unsigned int max (2^32-1):

    Print error message and assume value of 2021 (so *seedValue* = 2021)

If *seedValue* is less than unsigned int min (0 - indicates negative sign):

    Print error message and assume value of 2021 (so *seedValue* = 2021)

Else, continue with *seedValue* = user's input

[Creating arrays for the pig roll]

Create array of k amount of 0s (represents each player's initial points), call array *points*

Create another array that enumerates the 5 positions (side, jowler, razorback, trotter, and snouter), call it *Positions* (as stated in assignment document)

Create another array of the roll possibilities (side, side, jowler, jowler, razorback, trotter, and snouter), call it *pig* (as stated in assignment document)

[Simulating the pig roll]

Set srandom() to *seedValue* (similar to example in assignment document)

For each player, starting with 0th player and until the last k-1 player:

    Print corresponding player name from names.h and "rolls the pig"

    Randomly select value from *pig* with random()

    If pig[random() % 7] not equal to side:

        If pig[random() % 7] equals jowler:

            Print that the player rolled jowler

            Increment points[i] by 5

            If points[i] greater than or equal 100:

                Break

            Else:

                Continue (go back to the start of this for loop)

        If pig[random() % 7] equals razorback:

            Print that the player rolled razorback

            Increment points[i] by 10

If points[i] greater than or equal 100:

Break

Else:

Continue (go back to the start of this for loop)

If pig[random() % 7] equals trotter:

Print that the player rolled trotter

Increment points[i] by 10

If points[i] greater than or equal 100:

Break

Else:

Continue (go back to the start of this for loop)

If pig[random() % 7] equals snouter:

Print that the player rolled snouter

Increment points[i] by 15

If points[i] greater than or equal 100:

Break

Else:

Continue (go back to the start of this for loop)

If pig[random() % 7] equals side:

Print that the player rolled side

If i equals *k*-1:

Set i back to 0 (essentially starts a new round)

Else:

Increase i by 1 (moves to next player)

Go back to the start of the outermost for loop

[Concluding game]

Print name of winner and their number of points

End the main loop

Notes

- "random() % 7" makes sure that the randomly generated values are between 0 and 6 (where each number corresponds to a position in *pig* according to its index)
- Checking if the input is an integer might not be necessary (scanf should take care of this)
- Randomly selecting a *pig* value might be set up/placed differently in my actual code. Same goes for the if statement following it (I hope to come up with a more efficient method later)

Overall Description

The code I have in mind is structured so that my main function has everything I need. First, I will prompt the user to enter the number of players, making sure that it satisfies the requirements described in the assignment document. I will do the same with the random seed, prompting the user to enter a value and ensuring that it fits the necessary conditions.

Then, I will create the arrays that I need. The array called *points*, which will be initialized to have *k* amount of zeros, will keep track of each player's points as the game progresses. As the assignment document suggests, I will also enumerate positions and create an array called *pig* that represents the probability of each pig position (side, jowler, razorback, trotter, and snouter).

Next is the biggest portion of code: the pig rolling simulation. This would require setting up the random commands with my seed and then entering a loop where each player takes turns rolling. I figured a for loop would be best to keep track of each player's turn. It would allow me to alter the iterator and restart the round as needed. In this for loop, I will randomly select a value and have it represent the index of the *pig* array, thereby randomly selecting a pig position. I will check to see which of the five pig positions it is and then increment the points array accordingly (by 0, 5, 10, or 15 points). I will also check to see if the points for each player after each roll is greater than or equal to 100. If this is true, I will break the loop and state them as the winner. Otherwise, I will continue the loop until the winner is decided upon.

Simple Summary

- Function: main(void)
- Inputs: number of players and random seed
- Variables: *k* (number of players) and *seedValue* (random seed)
- Outputs: names of players as they take turns, which positions their rolls resulted in, and who ultimately won the game

- Process: prompt for both inputs, create arrays, simulate the pig rolls, and end the game

Goals/Intended Process

- Set up this pseudocode in C
- Address possible errors
- Swap loops for more efficient methods once running properly
- Add sufficient comments and clean up format