Pass the Pigs Design.pdf

Program Description

This program represents a game in which a certain number of players roll an asymmetrical dice, referred to as the pig. There are five different sides to the pig, and each side represents a different action taken. Four of the five sides grant the player a given number of points, while the other side ends the player turn. Each turn goes until the turn end side is rolled, or the player reaches a winning score. For the purposes of this assignment, the winning score is 100. The program takes two inputs, the number of players, and the seed for random number generation.

Files included

pig.c: Code that contains main() and the majority of the code for the simulation of the game

names.h: File that contains an array of length 10, with names of players.

Makefile: Formats program into clang format and compiles it into an executable.

README: File in markdown format that describes building and running the program

DESIGN.pdf: Describes the design for the program with pseudocode and illustrations.

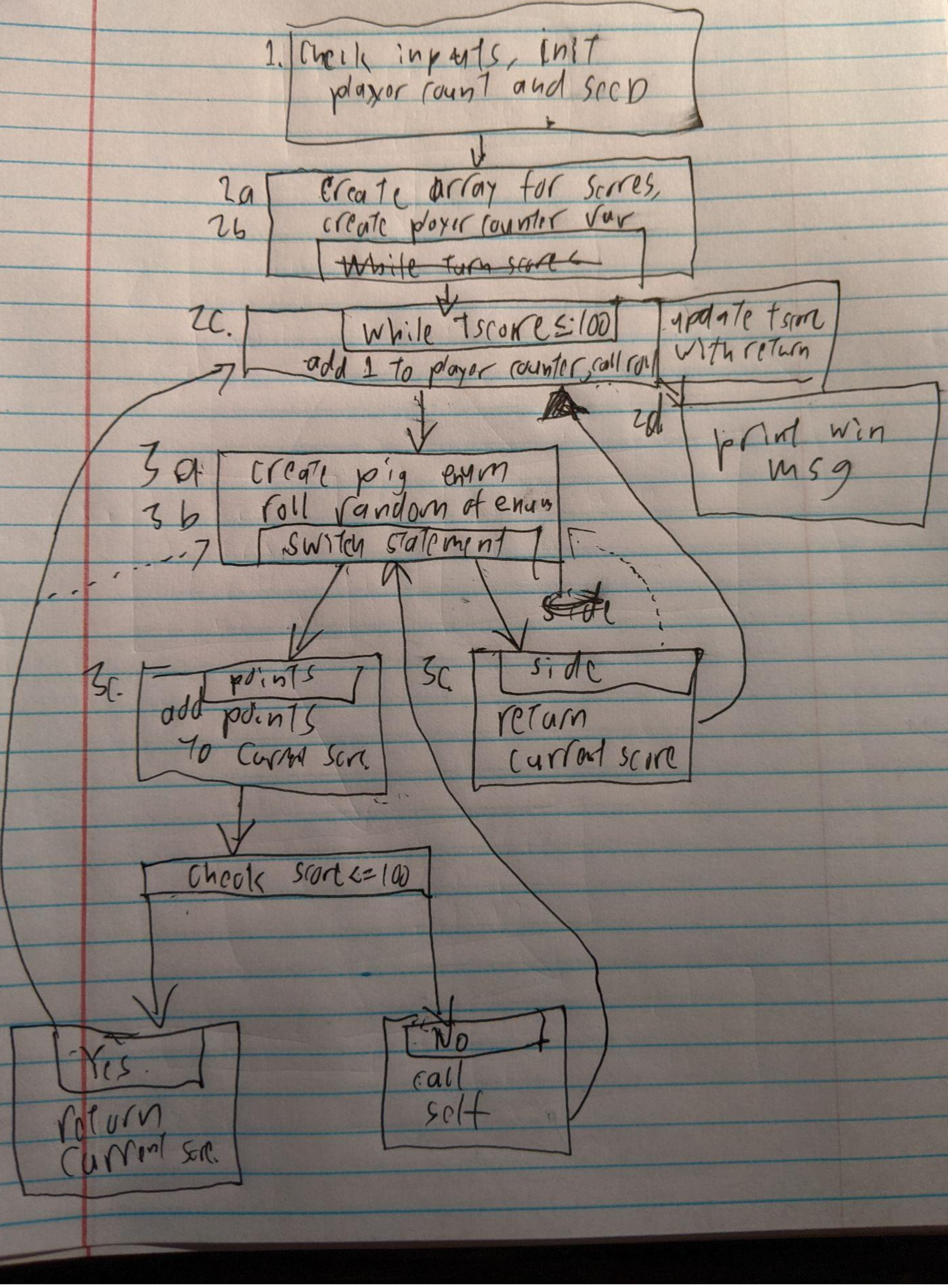
Pseudocode/Structure(Still in Progress):

Get user input for the number of players, between 2 and 10 inclusive. If input not specified, print an error message and use 2.
Get user input for random seed, between 0 and maximum value of unsigned integer, inclusively. If invalid input is given, use 2021 instead. The constraints on this input are to make sure seed is a valid input for the srandom function.

Numbers here are for numbers in diagram below.

```
1. Check inputs
    a. First check if first input is 2<=input<=10. If yes, set playercount as given int. If no,
       print error msg, then set 2
    b. Check second input if it is a positive int. If yes, use srand as seed as given int , if
       no, print error msg, then set seed as 2021.
2. Player turn selection/game state update
    a. This function takes playercount, and returns void
    b. create an int of current player number that starts at -1, and an int of turn score
       that starts at 0, and a score array of zeros called scores with length(playercount)
    c. While turnscore <=100
        i. If currentplayernum==playercount -1
            1. currentplayernum=0
            2. else:currentplayernum+=1
        ii. Print (names[currentplayernum], rolls the pig...)
        iii. Then call turn function (4.) current player number, and
             scores[currentplayernum]
```

            iv. Set turnscore = output of turn function
    d. After code exits loop, names[currentplayernum] wins with score scores[currentplayernum]
3. Player turn function
    a. curren playernum, and an int for current score return an int (of score)
    b. Roll random from 0 to 6, to do this, do random int modulo 7
    c. Do a switch statement for the 7 different sides of pig
        i. If any of the sides that yield points, print message that names[currentplayernum]rolls name of roll, add the appropriate score to current score
            1. If currentscore <= 100
                a. call self(4.) playernum, and updated current score
                b. Else: return currentscore
        ii. If side: return currentscore


1. Check inputs, init player count and seed
2a Create array for scores,
2b create player counter var
~~While turn score <~~
2c. While tscore ≤ 100
add 1 to player counter, call roll
update tscore with return
2d print win msg
3a create pig enum
3b roll random of enum
switch statement
side
3c. points
add points to current score
3c side
return current score
Check score <= 100
Yes
return current score
No
call self