Design Document - Pig.c

Summary of the program:
This program is supposed to simulate a game of Pigs. The game Pigs, is a dice game, where a number of players each take turns rolling a weighted die that simulates rolling a “pig”. Based on which side the die landed on, you would either get a certain number of points and get to reroll, or if it landed on the “side”, you would not gain any points and forfeit your turn to the next player. First player to reach 100 or above points wins. Each side of the die had a different probability of landing, which means that it would be a more complex game of chance. The point system was arranged as followed:

Orientation	Point Value	Probability
Side	+0 & Lose Turn	2/7
Razorback	+10	1/7
Trotter	+10	1/7
Snouter	+15	1/7
Jowler	+5	2/7

The program would ask for 2 inputs, one would be the number of players playing the game (from 2 - 10), and what seed the players would like to use (from 0 - UINT_MAX). If the player did not input a valid number of players, the program would use a default value of 2 players, and for an invalid seed, the program would use a default value of 2021 (Probably because we are in the year 2021, get it?).

How to Run the Game:
To run this program, you may compile the program first to make sure everything is in order. This is simple since we have a MakeFile with the game, where all you have to type is:

```
$ make
```

Otherwise, you can use the program:

```
$ clang -Wall -wextra -Wpedantic -o pig pig.c
```

The command to run is:

```
$ ./pig
```

You will then be prompted on entering a number of players between 2 and 10. Once imputed, you will then be prompted to enter a valid seed, between 0 and UINT_MAX (Unsigned Integer Max). The program will then print out each player round that is played, and each permutation that is rolled, finalizing with which player ended with a score of 100 or over.

Pseudocode:
Include names file

Include limits
Include standard io
Include standard library

Define a new enumerated type called Position (SIDE, RAZORBACK, TROTTER, SNOUTER, JOWLER)
Create pig array with different positions with probability on how many times they can appear

Create points_and_wincondition function that takes as input points array, won point value, and current player, adds points to associating player in point array and tests to see if their number of points is equal to or above 100 and if so return 1, and if not, return 0 (easier down the line in the actual game)

Main:

Ask how many players (use scanf, check if input is less than 2 or greater than 10, if so set numplayers to 2
If in range set it to numplayers

Ask for random seed (long type) (use scanf, check if input is less than 0 or greater than UNIT_MAX, if so set unsigned type seed to 2021
If in range set it to unsigned and put it into seed
(Makes sure to check if the output of scanf is 0, and if it is print error and set to default)

Set up points array length of numplayers and all values 0

Set srandom with seed
Wincondition = 0
Currentplayer = 1
while(true)
    Roll = 1
    print(currentplayer name is rolling)
    while(roll)
        Set randomposition to enumerated side using random%7
        If(lands on back)
            Prints lands on back
        Else
            Prints lands upright
        if(points_and_wincondition function)
            Wincondition = 1
            Break

```
        If else(lands on snout)
                Prints land on snout
                if(points_and_wincondition function)
                        Wincondition = 1
                        Break
        If else(lands on ear)
                Prints land on ear
                if(points_and_wincondition function)
                        Wincondition = 1
                        Break
        Else
                Prints land on  side
                if (current player equals numplayers -1)
                        Currentplayer = 0 (end of last player round goes to first player round)
                Else
                        Currentplayer + 1
If(wincondition equals 1)
        Break
Prints currentplayer won with associating number of points in points array
```

General Structure of the Code and How It Works:

Essentially the code is written in 3 parts. The first part is to check user input for the number of players. Conditions are set such that the number of players is only limited as little as 2 or as large as 10. Once this condition is met, it sets the input to a saved variable we will use later. If it fails this condition, it instead prints an error and uses the set default of 2 players.

The second part of the code is the seed input. The user is prompted to input a seed, which it checks with a condition. If the seed is smaller than 0, or if it is larger than UINT_MAX defined by the limits library, it uses a set default seed of 2021 and makes it equal to a variable that we will use later. If the seed is in range, it uses the imputed seed instead. There is also another check that I wrote at the beginning of the part, which checks to see if the output of scanf is 0. If it is, which means that there were 0 successful transmissions from scanf (no ints were able to be read), it prints the error message and sets the default seed to 2021.

The last part is the actual code for the game. It includes two nested while loops, the first one mimics each player round of the game, and the second round mimics each roll from the current player. It checks to see what permutation the current player rolled, and runs the function I created at the beginning of the file, which adds the number of correct points to the player's position in the points array, and checks to see if the player has at least 100 points. If so, the function outputs 1, which then breaks out of the second while loop, and eventually with the win condition set to 1 from the function, it breaks from the first while loop, and prints the winning statement with how many points they had. If the player rolls a side, it sets the current player to player++ or to the first player if it was the

last player to roll a side. It then exits the second while loop and restarts the first while loop.

Image:
Here is an image that depicts the game. We start with the first player, they roll to get points based on the permutation of the pig until they get a side. Once they get a side, the pig gets transferred to the next person, and so on. Once a player gets 100 points or above, they win.