Rules and Document break down:

K players, such that K is greater than or equal to 2 or less than equal to 10

There are 5 possible ways for the pig to land:

1. **Side**: Pig lands on the side (either left or right side) [2/7]
2. **Razorback**: Pig lands on its back [1/7]
3. **Trotter**: Pig lands upright [1/7]
4. **Snouter**: Pig lands on its snout [1/7]
5. **Jowler**: Pig lands on one of its ears [2/7]

Rolling **Side** yields 0 points and immediately ends the current player's turn

The other player then rolls the pig

Game always start at player 0

Rolling either **Razorback** or **Trotter** earns 10 points for the player. Rolling Snouter earns 15 points. Lastly, rolling Jowler 5 points.

The game ends when any player has earned 100 or more points

Enumeration:

Use enumerations to represent each of the positions.

Enum are used to provide names for integer constants. Using this, we can represent the positions and the pig in the following manner

```
typedef enum { SIDE, RAZORBACK, TROTTER, SNOUTER, JOWLER } Position;
const Position pig[7] = {
  SIDE,
  SIDE,
  RAZORBACK,
  TROTTER,
  SNOUTER,
  JOWLER,
  JOWLER
};
```

The typedef is used to give a new name to a type. In this case, we used typedef to name the enumeration of positions as Position. The pig, then, can be represented as an array of positions.

The act of “rolling” the pig can be achieved by randomly selecting 1 of the 7 elements of the pig array.

Pseudorandom Numbers (simulate the rolling of pig):

Need *pseudo random number generator* (PRNG)

Utilize srandom() and random()

after calling srandom() with a seed to set, that the pseudorandom numbers that are generated with random() always appear in the same order.

Used so we can check if our program matches with the output of the program’s example

<u>What I need to do for this task:</u>

1. Ask the user to input the number of players, scanning input from stdin. Use scanf() to ask for input. If the player inputs a value that is between 2 and 10 inclusive, print →

```
1 fprintf(stderr, "Invalid number of players. Using 2 instead.\n");
```

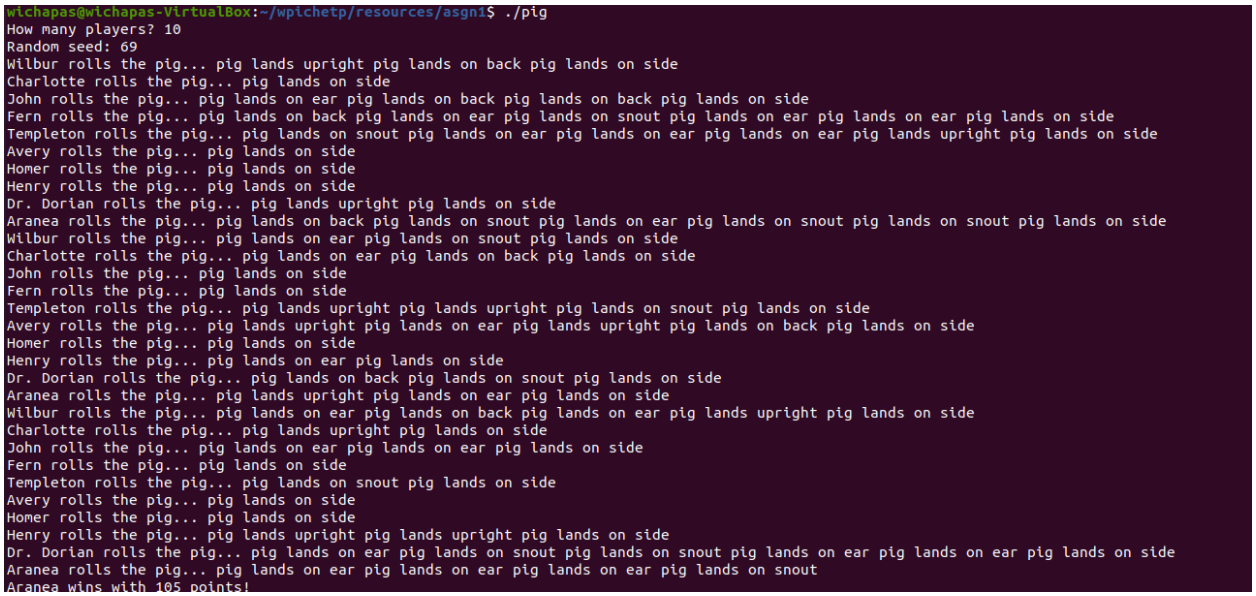
2. Ask the user to input the random seed. If user inputs anything other than valid seed, print error like this →

```
1 fprintf(stderr, "Invalid random seed. Using 2021 instead.\n");
```

   then use 2021 as the default random seed
3. Set the random seed and make sure each player starts at 0 points.
4. Proceed around the circle starting from player 0. For each player:

a) Print out the name of the player currently rolling the pig. Use names.h file. Index 0 is player 0 etc
b) Roll the pig, increase the player’s point count until they either win the game or the pig lands on one of the 2 sides
c) If the player has greater or equal to 100 points, they win the game and a congratulatory message is printed to stdout
d) If rolled pig lands on one of its two sides, the player’s turn ends and the next player in the circle gets to roll

Expected input/output



Easily, the first few part of the code will be the set up of taking in inputs of the number of players and the random seed. Then I have to create a loop that will allow the player to keep rolling until they roll a side, which a new player will be rolling instead. A few counters needed to keep track of the player rolling. Also, keeping track of the points can easily be done with a list or an array. To break out of the loop, check if some player has won 100 or more points. Then, we print the winner.

Rough Pseudocode:

player_points = [] (this is like an array or somethn)
player_counter = 0
Names.h = {player names here}

input1 = ask for user to input number of players
input2= ask for user to input random seed

numPlayers = input1

Current_player_formula = current_player % input1

Print first player name
    While true…

        Roll random dice
        current_player_formula

If result == “**Razorback” OR “Trotter”**:
player_points[player] += 10
Is result 100 or more?
Winner found!
continue

If result == “Snouter”:
player_points[player] += 15
Is result 100 or more?
Winner found!
continue
If result == “Jowler”:
player_points[player] += 5
Is result 100 or more?
Winner found!
continue

If result == “Side”:
Player_counter += 1
Recalculate current player
continue
WinnerFound:
Print “winner found! With ___ points”

Exit program

Explanation and Thought Process:

The set up:

After I have read the rules of the game and the materials that I was given for this assignment, I kind of had a rough idea of what needed to be done. First, I need a counter to keep track of the current player in order to use it in conjunction with the names.h file. Next, I also needed a way to store the score of each player so I chose to use an array. I allocated the space of 10 elements as the limit of the number of players is 10. We can simply rotate through the array by using the current_player counter.

The loop:

The while loop is constructed with the rolling of the dice and printing the player name at the top. This is because as seen in the example output of the test file, we have to construct a loop where a player must keep rolling the dice, adding up points, until you land on side. Once that happens, the player_counter increases and switches over to the next player. The same process then repeats. Once a pig has been rolled and checked, continue will send us to the top of the loop to reroll the dice.