Fabrice Kurmann
fkurmann@ucsc.edu
7 April, 2021

CSE 13S Assignment 1: Left, Right and Center Design Document

Assignment: Create a simulator in C for the game left right and center (rules below) which reads users input for a starting seed and player number.

Rules of the game:

K players, K between 1 and 14 inclusive.

Each player starts with 3$.

Three dice each with:
- 3/6 outcome of DOT
- 1/6 outcome of L
- 1/6 outcome of R
- 1/6 outcome of C

Game Flow:

Player 1 rolls:
- if 3$ or more, roll dice 3x
- if 2$, roll dice 2x
- if 1$, roll dice 1x

- if dice = L, give 1$ to player on left
- if dice = R, give 1$ to player on right
- if dice = C, put 1$ into center
- if dice = dot, do nothing

Move onto player to the right: Players are numbered 1-x, to move from player to player ** See Appendix

Play until only one person has all money remaining.

Finally return money in pot and money held by last man standing.

Program Arguments:
- -Random seed, so that the random # is always the same on all computers it is run on
- -Number of players

For testing: Due to this random seed, same inputs should always yield same outputs.

Note: Arguments will be gotten through a scanner at the beginning of the program.

main() has no other arguments

Information to Store:
- -Original number of players, original_num_players (from scanner)

-Real time number of players, num_players
-Which player is playing currently, currentPlayer (integer value)
-Amount of money in center (integer value)

-Amount of money each player has (array corresponding to each player)
-Name corresponding to each player (external array in file philos.h)

-Player to the left, found using helper method left (integer value)
-Player to the right, found using helper method right(integer value)

Game Pseudocode:

```
main()

// Setup phase
Random_seed = ReadInput
If Random_seed == negative:
    invalid argument message, end program

Num_players = Original_Num_Players = ReadInput
If Random_seed != 0<r<14:
    invalid argument message, end program

CurrentPlayer = 0

// Play Phase
while (numPlayers > 1):
    //Run the game as long as there are multiple players
    numRolls = 0 //every loop, reset this variable

    // Find number of rolls a player gets
    if money(currentPlayer)>0:
        if money(currentPlayer) == 1:
            numRolls = 1
        if money(currentPlayer) == 2:
            numRolls = 2
        if money(currentPlayer) >= 3:
            numRolls = 3
    else: //move onto the next player
        break

    find left and right players with helper method
```

Note on unexpected challenges in play cases:
During the play phase, a significant oversight in my first design plan that only became apparent in the debugging phase was that both during (between turns) and after the last turn, the program would have to check whether the current player's balance slips to zero. Furthermore, the program has to check whether through giving money, a player resurrects another player. These checks have been added in their required spots below.

```
    // Roll and respond accordingly
    for (numRolls):
```

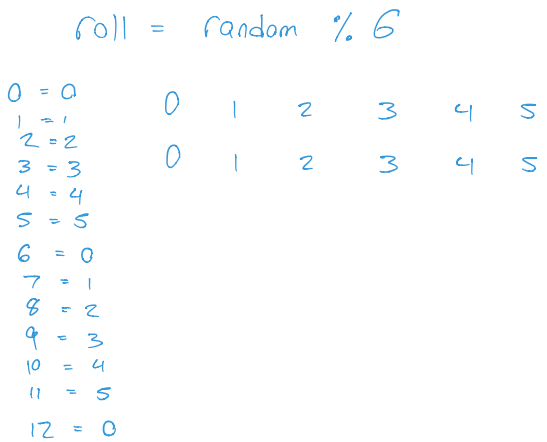
```
if money(player) == 0: //scenario where player hits zero during turn
        numPlayers—
        break

        roll = random (random, seed)
        roll = roll % 6 *

        if roll == left:
                if money(playerleft == 0):
                        numPlayers ++
                transfer money to player to left in array **
        if roll == right:
                if money(playerright == 0):
                        numPlayers ++
                transfer money to player right in array **
        if roll == center:
                transfer money to center from array,
        else: pass
if money(player) == 0: //scenario where player hits zero in last roll of turn
        numPlayers—
        break
player = playerright
```

Appendix

*Roll will create any positive integer using the random function, how to use mod to assign values only 0 to 5:



**Player Numbering, using given mod based functions, this is and visual example of how the players to the left and right an be found. Algorithm works exactly as needed.