Assignment 1
Left, Right and Center

Prof. Max Dunne
CSE 13S – Fall 2020

1 Introduction

We are going to implement a simple game called *Left, Right, and Center*. It requires no skill, no real decision-making, and a player that is out of the game can suddenly come back in and often win.

Some of you may have religious or moral prohibitions against gambling. This program is not gambling, since (i) you neither win nor lose, and (ii) only fictitious people lose or win any money. But, if you do have any qualms, let us know and we will give you an alternative assignment.

2 Playing the Game

Some number of k players, $1 < k \leq 10$, sit around a table. Each player has in her hand \$3. There are three dice, and each die has 6 faces and is labeled: $3 \times \bullet$, $1 \times$ **L**, $1 \times$ **R** or $1 \times$ **C**. As a result, we know that there is a 50% chance of rolling $\bullet$, and $16.6\overline{6}\%$ chance of rolling each of **L**, **R**, or **C**.

1. Beginning with player 1, roll the dice:
   (a) If the player has \$3 or more then she rolls three dice; if she has \$2 then she rolls two dice; if she has only \$1 then she rolls one die; if she has no money then she must pass.
   (b) For each die:
      i. If the player rolls **L** then she gives \$1 to the player on her *left*.
      ii. If the player rolls **R** then she gives \$1 to the player on her *right*.
      iii. If the player rolls **C** then she puts \$1 in the pot in the *center*.
      iv. If the player rolls $\bullet$ then she ignores it.
2. Move to the next player in sequence: to the right. The players are numbered. There is `you`. Then there is the left player which is `(you - 1) mod` the number of players and there is the right player which is `(you + 1) mod` the number of players. Be careful: What does `-2 % 10` mean? Consequently, you may find this code useful:

```
//
// Returns the position of the player to the left.
//
// pos:     The position of the current player.
// players: The number of players in the game.
//
uint32_t left(uint32_t pos, uint32_t players) {
  return ((pos + players - 1) % players);
}

//
// Returns the position of the player to the right.
```

```
//
// pos:     The position of the current player.
// players: The number of players in the game.
//
uint32_t right(uint32_t pos, uint32_t players) {
  return ((pos + 1) % players);
}
```

3. Repeat until only one player has any money remaining (who then wins the pot).

3 Your Task

- You must have one source file: `lrc.c`. Do not name your source file anything else. You will lose points.
- For grading purposes the numbering of the faces matters, and so they must be defined as follows (do not change it):

```
typedef enum faciem {LEFT, RIGHT, CENTER, PASS} faces;
faces die[] = {LEFT, RIGHT, CENTER, PASS, PASS, PASS};
```

  This means you may not use `int` as a substitute.

- You must give your players names, and for grading purposes the names must correspond to these (do not change them):

```
const char *names[] = {"Happy", "Sleepy", "Sneezy", "Dopey",
                       "Bashful", "Grumpy", "Doc", "Mirror Mirror",
                       "Snow White", "Wicked Queen"};
```

- Typing `make` must build your program and `./lrc` must run your program. Since you have not learned about `Makefiles` yet, here is one that you can use for now.

```
CFLAGS=-Wall -Wextra -Werror -pedantic
CC=clang $(CFLAGS)

lrc     :       lrc.o
        $(CC) -o lrc lrc.o
lrc.o   :       lrc.c
        $(CC) -c lrc.c
clean   :
        rm -f lrc lrc.o
infer   :
        make clean; infer-capture -- make; infer-analyze -- make
```

Makefile

- You will need to use a random number generator to simulate rolling a dice. You can do this by calling the function `rand()` (read the *man page*) to get a random value. You can then use `mod` to limit this value to the range of 0–5 inclusive (in Computer Science, we start from 0). This value is used to determine what was rolled.
- In order that your program be *reproducible*, you must start from a known place. This is accomplished by *setting the random seed* using the function `srand()` (again read the *man page*). Your program will ask for two numbers: the random seed and the number of players. You should assign these inputs to variables to

use in your program. The random seed completely determines the outcome of your program. If you give it the same random seed and the number of players you *must* get the same answer. Here is an example of using `srand()` and `rand()`:

```
srand(1); // This sets your random seed to 1.
int a = rand(); // Declares & initializes variable to random number.
```

- *Comment your code.* Include block comments to tell the grader what a certain block of code does. Use a line comment to explain something that is not as obvious. Refer to the coding standards.
- Your program must use `clang-format`.
- Your program *must* run on the time share. If it does not, your program will receive a 0. To avoid this, test your program on the time share.
- Your program must pass `infer` with no errors. If there are any, fix them; for those that you cannot fix, make sure to document them in your `README`. `infer` is installed on the school servers, where you should be testing anyway.

In lecture we talked briefly about *random* and *pseudo-random* numbers. As we know, computers produce pseudo-random numbers, and in this case it is to your benefit since *reproducibility* is essential. That means that in reality you program though it appears to be random is actually *deterministic*. This is why starting with the same seed produces the same sequence.

4 Hints

1. *Start Now!* You can always finish early.
2. You may find it helpful to draw out and run through a game to get a feel for the rules. Doing so may make it easier to visualize the game and design your program.
3. The game itself should be an infinite loop, where the condition to break out of this loop is when there is one active player remaining.
4. You should think carefully about what quantities that you must track in order for your program to function. At a *minimum* you must keep track of the bank balance of each player, the amount of money in the pot, and the number of players that are *in*. Be careful: players that were *out* may be brought back in if money is passed to the *left* or *right*.

5 Deliverables

You will need to turn in:

1. `lrc.c`: The source file which is your program.
2. `Makefile`: This is a file that will allow the grader to type `make` to compile your program. Typing `make` with no arguments must build your program.
   - `CFLAGS=-Wall -Wextra -Werror -Wpedantic` must be included.
   - `CC=clang` must be specified.
   - `make clean` must remove all files that are compiler generated.
   - `make` should build your program, as should `make all`.
   - Your program executable must be named `lrc`.

3. `README.md`: This must be in markdown. This must describe how to use your program and `Makefile`.

4. `DESIGN.pdf`: This *must* be a PDF. The design document should describe your design for your program with enough detail that a sufficiently knowledgeable programmer would be able to replicate your implementation. This does not mean copying your entire program in verbatim. You should instead describe how your program works with supporting pseudo-code. For this program, pay extra attention to how you describe your logic flow/algorithm in **C**.

All of these files must be in the directory `asgn1`.

6 Submission

To submit your assignment, refer back to `asgn0` for the steps on how to submit your assignment through `git`. Remember: *add, commit,* and *push*!

Your assignment is turned in *only* after you have pushed. If you forget to push, you have not turned in your assignment and you will get a *zero*. "I forgot to push" is not a valid excuse. It is *highly* recommended to commit and push your changes *often*.

7 Supplemental Readings

- *The C Programming Language* by Kernighan & Ritchie
  - Chapter 1 §1.6, 1.7
  - Chapter 2 §2.3
  - Chapter 3 §3.1-3.7

Example

Note that due to version differences there is no guarantee your example will be exactly the same. This is for the overall flow of the game.

```
-bash-4.2$ ./lrc
Random seed: 2020
How many players? 7
Happy rolls... gets a pass gets a pass gets a pass
Sleepy rolls... gets a pass gets a pass gets a pass
Sneezy rolls... gets a pass gives $1 to Sleepy gives $1 to Dopey
Dopey rolls... puts $1 in the pot gets a pass puts $1 in the pot
Bashful rolls... gives $1 to Grumpy puts $1 in the pot gets a pass
Grumpy rolls... puts $1 in the pot puts $1 in the pot gives $1 to Doc
Doc rolls... puts $1 in the pot puts $1 in the pot gets a pass
Happy rolls... gets a pass puts $1 in the pot gives $1 to Sleepy
Sleepy rolls... gets a pass gets a pass gets a pass
Sneezy rolls... gets a pass
Dopey rolls... gets a pass gives $1 to Sneezy
Bashful rolls... puts $1 in the pot
Grumpy rolls... puts $1 in the pot
Doc rolls... gets a pass gives $1 to Happy
Happy rolls... gets a pass gets a pass
Sleepy rolls... gets a pass gets a pass gets a pass
Sneezy rolls... gives $1 to Sleepy gives $1 to Sleepy
Dopey rolls... gives $1 to Bashful
Bashful rolls... gives $1 to Grumpy
Grumpy rolls... gives $1 to Doc
Doc rolls... gets a pass gets a pass
Happy rolls... gives $1 to Sleepy gets a pass
Sleepy rolls... gets a pass gets a pass gets a pass
Doc rolls... puts $1 in the pot gives $1 to Grumpy
Happy rolls... gets a pass
Sleepy rolls... gives $1 to Sneezy gets a pass gives $1 to Happy
Sneezy rolls... gets a pass
Grumpy rolls... gets a pass
Happy rolls... gets a pass gives $1 to Sleepy
Sleepy rolls... gives $1 to Sneezy gives $1 to Happy gets a pass
Sneezy rolls... gets a pass gets a pass
Grumpy rolls... puts $1 in the pot
Happy rolls... gives $1 to Doc gets a pass
Sleepy rolls... gets a pass gets a pass puts $1 in the pot
Sneezy rolls... gives $1 to Sleepy gives $1 to Sleepy
Doc rolls... puts $1 in the pot
Happy rolls... gets a pass
Sleepy rolls... puts $1 in the pot gets a pass gives $1 to Sneezy
Sneezy rolls... gives $1 to Sleepy
Happy rolls... gets a pass
Sleepy rolls... gets a pass gives $1 to Happy gets a pass
Happy rolls... puts $1 in the pot puts $1 in the pot
Sleepy wins the $17 pot with $4 left in the bank!
```