Assignment 1
Left, Right and Center

Prof. Darrell Long
CSE 13S – Spring 2021

Due: April 11th at 11:59 pm

1 Introduction

The gambling known as business looks with austere disfavor upon the business known as gambling.

—Ambrose Bierce

We are going to implement a simple game called *Left, Right, and Center.* A subset of philosophers will represent players of the game. It requires no skill, no real decision-making, and a philosopher 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 involving vampires.

2 Playing the Game

No sympathy for the devil; keep that in mind. Buy the ticket, take the ride. . . and if it occasionally gets a little heavier than what you had in mind, well. . . maybe chalk it off to forced conscious expansion: Tune in, freak out, get beaten.

—Hunter S. Thompson, *Fear & Loathing in Las Vegas*

Some number of k players, $1 < k \le 14$, 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 •, 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 • 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:

```c
//
// Returns the position of the player to the left.
//
// pos:     The position of the current player.
// players: The number of players in the game.
//
static inline uint8_t left(uint8_t pos, uint8_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.
//
static inline uint8_t right(uint8_t pos, uint8_t players) {
  return ((pos + 1) % players);
}
```

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

3 Your Task

Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.

—John von Neumann, 1951

- 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):

```c
typedef enum faciem { PASS, LEFT, RIGHT, CENTER } 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):

```c
char *philosophers[] = { "Immanuel Kant",
                         "Martin Heidegger",
                         "David Hume",
                         "Georg Wilhelm Friedrich Hegel",
                         "Arthur Schopenhauer",
                         "Ludwig Wittgenstein",
                         "Karl Wilhelm Friedrich Schlegel",
                         "Friedrich Nietzsche",
                         "Socrates",
                         "John Stuart Mill",
                         "Plato",
                         "Aristotle",
                         "Thomas Hobbes",
                         "Rene Descartes" };
```

This array of names is defined in `philos.h`, which can be found in the course resources repository.

- 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.

Makefile

```make
CC = clang
CFLAGS = -Wall -Wextra -Werror -Wpedantic

all: lrc

lrc: lrc.o
	$(CC) -o lrc lrc.o

lrc.o: lrc.c
	$(CC) $(CFLAGS) -c lrc.c

clean:
	rm -f lrc lrc.o

scan-build: clean
	scan-build make
```

- You will need to use a random number generator to simulate rolling a dice. You can do this by calling the function `random()` (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 `srandom()` (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 `srandom()` and `random()`:

```c
srandom(1);        // Sets the random seed as 1.
int a = random(); // Set a as a pseudorandom 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.
- All source code should be formatted using `clang-format`.
- Your program *must* compile and run on Ubuntu 20.04 with no errors. If it does not, your program will receive a 0.
- Your program must pass `scan-build` with no errors. Document any errors that are false positives in your `README.md`.

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 Resources

Common sense in an uncommon degree is what the world calls wisdom.

—Samuel Taylor Coleridge, *Literary Remains*

A working example binary can be found in the course resources repository on `git.ucsc.edu`. The header file, `philos.h`, containing the names of the philosophers can be found here as well.

For this assignment, running "`./lrc`" will prompt the user to enter a random seed and the number of players. It will then produce an output of the game which your program should also be able to produce given the same seed and number of players. Your program must be able to do *exactly* the same.

5 Hints

It is no use trying to sum people up. One must follow hints, not exactly what is said, nor yet entirely what is done.

—Virginia Woolf

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 only 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*.

6 Deliverables

The design process is about designing and prototyping and making. When you separate those, I think the final result suffers.

—Jonathan Ive

You will need to turn in:

1. `lrc.c`: The source file which is your program.
2. `philos.h`: The header file containing the names of the philosophers.
3. `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.
   - `CC=clang` must be specified.
   - `CFLAGS=-Wall -Wextra -Werror -Wpedantic` must be included.
   - `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`.
4. `README.md`: This must be in markdown. This must describe how to use your program and `Makefile`.
5. `DESIGN.pdf`: This must be in PDF format. 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**. The commit ID containing the first draft of your design document must be submitted no later than the Thursday before the assignment is due, by 11:59 pm PST. Failing to submit this first draft or submitting a insubstantial draft means receiving a 0 for your design document.

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

7 Submission

Better three hours too soon than a minute too late.

—William Shakespeare

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*.

8 Supplemental Readings

I was not proud of what I had learned but I never doubted that it was worth knowing.

—Hunter S. Thompson, *The Rum Diary*

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

*La programmation en **C** c'est comme donner une tronçonneuse à un singe.*