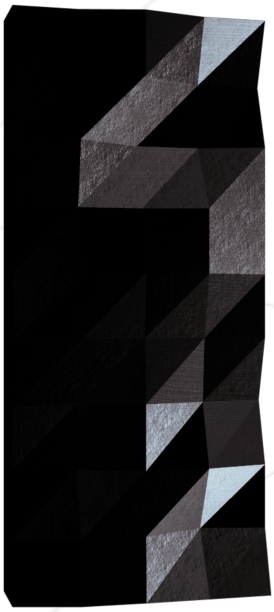

Dash

Rock, Paper, Scissors, Lizard, Spock

quartz stephen@42.us.org

Summary: Taste the const poison.

Contents

Chapter I

Foreword

Scene: The apartment.

Sheldon: Oh look, Saturn 3 is on.

Raj: I don't want to watch Saturn 3. Deep Space Nine is better.

Sheldon: How is Deep Space Nine better than Saturn 3?

Raj: Simple subtraction will tell you it's six better.

Leonard: Compromise. Watch Babylon 5.

Sheldon: In what sense is that a compromise?

Leonard: Well, five is partway between three... Never mind.

Raj: I'll tell you what, how about we go rock-paper-scissors?

Sheldon: Ooh, I don't think so. No, anecdotal evidence suggests that in the game of rock-paper-scissors, players familiar with each other will tie 75 to 80% of the time due to the limited number of outcomes. I suggest rock-paper-scissors-lizard-Spock.

Raj: What?

Sheldon: It's very simple. Look, scissors cuts paper. Paper covers rock. Rock crushes lizard. Lizard poisons Spock. Spock smashes scissors. Scissors decapitates lizard. Lizard eats paper. Paper disproves Spock. Spock vaporizes rock. And as it always has, rock crushes scissors.

Raj: Okay, I think I got it. *(They prepare)*

Together: Rock-paper-scissors-lizard-Spock! *(Both hold up the symbol for Spock)* Oh!

Chapter II

Introduction

Dashes are projects available only for a limited amount of time. You can access it if you are present at the school and logged in. You will have little time to complete it, and the clock starts ticking the moment the repository is created. After the time is up, your code will be evaluated by moulinette. If you succeed, you will earn a little XP for your effort.

For this dash, you'll be rigging a game of `Rock-paper-scissors-lizard-Spock`, which is an expansion of the popular hand game Rock-paper-scissors. It plays as Sheldon describes in the foreward.

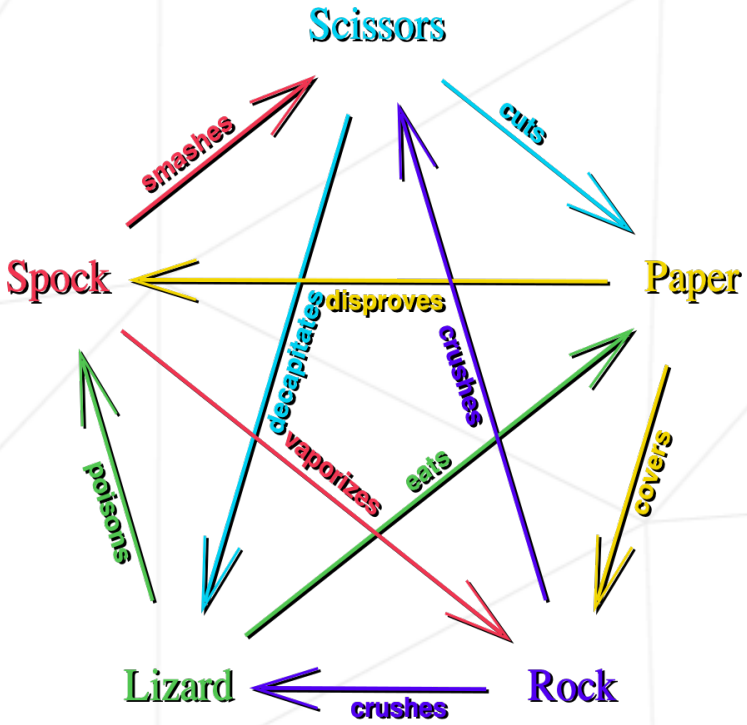

Figure II.1: Winning! Starting to get the concept now?

Chapter III

General instructions

- You must only submit one file: `rpsls.c`
- This dash must be written in C. The Norm isn't mandatory, but it is a good idea.
- Your goal is to rig the game so that you'll win **100%** of the `RPSLS` rounds.
- The only thing your function should be changing are **pointers**.
- There cannot be a `main` function in your submission.
- You have to handle errors carefully. In no way can your program, or in this particular case your function, quit in an unexpected manner (Segmentation fault, bus error, etc).
- The `libc` functions allowed on this project are `srand` and `rand`.

Chapter IV

Mandatory part

- You have to write a `rig_game` function that will modify an array of `moves` passed as one of its parameters so that you'll win every round. It must be prototyped as follows:

```
void rig_game(const char *const *moveset, const char **moves, int seed);
```

- `moveset` is a null-terminated array of strings always containing the following `moves`, in no specific order:
  - `"rock"`
  - `"paper"`
  - `"scissors"`
  - `"lizard"`
  - `"spock"`
- `moves` is a null-terminated array of pointers, each of which point to a `move` from `moveset`. These `moves` will be played in order (starting from index `0`) against your opponent for each round of `RPSLS`.
- `seed` is the seed used by your opponent to select a `move` from `moveset` to play against you for each round. It will select the `move` it plays as so:

```
const char *move = moveset[rand() % 5];
```

Be careful not to bring in any "outside" moves. Each move should be a pointer to a move from the moveset--not some rando string literal!

Chapter V

Turn-in and peer-evaluation

Turn in your 'dash' to the associated `Git` repository, as usual. It will be entirely evaluated by moulinette.