Assignment 1 – LRC Report Template

Jason Maheru

CSE 13S – Winter 24

Purpose

This program is a, simplified, simulation of the dice game called Left, Right, and Center. The game uses a single die, making the game entirely based on chance. Three to ten players are required to play.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

Randomness

Describe what makes randomness. Is it possible for anything to be truly random? Why are we using pseudorandom numbers in this assignment?

- Randomness is the lack of being able to predict something or being able to see the pattern in events. Something like flipping a coin can be considered truly random. We are using pseudorandom numbers because they are a way you can use randomness in a computational task, but can only be predicted if you know the seed value.

What is an abstraction

When writing code, programmers often use "abstractions". Define an abstraction in non computer science terms (Don't google it!)

- An abstraction is looking at the main details that are required and not the small details though. You take a complex concept and focus on the important characteristics.

Why?

The last assignment was focused on debugging. How can abstractions make debugging easier? What other uses are there for abstractions? Hint: Do you have to be the one to write the abstraction?

- Abstractions make debugging easier because abstractions layout to you for what you should be testing. Abstractions can help you realize what parts could be similar so when you start coding you'll already know what to use and you are just saving time.

Functions

When you write this assignment, you can chose to write functions. While functions might make the program longer, they can also make the program simpler to understand and debug. How can we write the code to use 2 functions along with the main one? How can we use 8 functions? Contrast these two implementations along with using no functions. Which will be easier for you? When you write the Program design section, think about your response to this section.

- **For two functions:** I would use one function for generating pseudorandom numbers. An input of the seed will be taken which the function would take to generate the pseudorandom number. The second function is to ask how many players are going to be playing.

- **For eight functions:** I would do the first 2 from the response above. The third function is for calling the names from the file. The fourth is for rolling the die. The fifth one is for initializing chips. The sixth one is for running the game itself. The seventh is for printing out each turn. The last one would be the main function itself.

- I will probably use the 8 functions because it will help me break up the code and make it easier to code since I am working on small parts at a time.

Testing

The last assignment was focused on testing. For this assignment, what sorts of things do you want to test? How can you make your tests comprehensive? Give a few examples of inputs that you will test.

- Test 1: Number of player input. If a more than 10 or less than 3 is inserted the code should present an error and run for 3 players. Also if I present a non-integer.

- Test 2: Test the pseudorandom numbers with a valid number and one with an invalid number.

- Test 3: Check for the error handling.

Putting it all together

The questions above included things about randomness, abstractions and testing. How does using a pseudorandom number generator and abstractions make your code easier to test?

- When using a pseudorandom number generator, if you know the seed then you will know what the output would be because every time it will be the same. So with that, you can control the test cases. With abstractions such as it is very obvious what your goals/outputs should be (knowing your requirements) you can then easily make test cases for the intended abstractions.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags that your program uses, and what they do.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[1]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like

[1] This is my footnote.

- The program will start by first asking: `"Number of players (3 to 10)? "`. You then input your answer. If you input anything out of the range of the given numbers then the program will automatically run with the default number of players which is 3. Then the program will ask you `"Random-number seed? "`. You will then enter a seed number. If the number is not valid then the program will use the default programmed seed number. Finally, the game will run and tell you how many tokens the player has after their turn and end the game when there is only one person left with tokens.

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

- The data structures used would be enumerations for the die sides. Arrays for the names and the positions of the die (face). The main algorithms: the game initialization, pseudorandom number generator, the player turns, check for winner, and error handling.

Pseudocode

Give the reader a top-down description of your code! How will you break it down? What features will your code have?

```
"#all include initiations"
typedef enum { DOT, LEFT, CENTER, RIGHT } Position;
const Position die[6] = { DOT, DOT, DOT, LEFT, CENTER, RIGHT };

void prng(unsigned seed);
     srandom(seed);

int promptNumberofPlayers(void)
     int num = 3
     print statement
     int result
     error statement
     return numplayers;

unsigned promptSeed(void)
     unsigned seed = 4823
     print statment
     int scanfresults
     print error code if < 1

void startingChips(int playerchips[], int numplayers)
     for each num of players
         playerchips[i] = 3

int rollDice(void)
     int ran = 0
     ran = random()% 6
     return ran

int checkWinner(int playerchips[], int numplayers)
     int winner = -1
```

```
    for int i=0, i < numplayers; ++1
        if playerchip[i] > 0
            if winner == -1
                winner = i
            else
                return -1

void turn(int playerchips[], int playerindex, int numplayers
    if playerchips[playerindex] > 0)
        int maxTurns = playerchips[playerindex] < 3 ? playerchips[playerindex] : 3;
        for (int i = 0; (i < maxturn)&& (playerchops[playerindex] > 0); ++i)
            int roll = rollDice()
    implement DOT, LEFT, RIGHT, CENTER using switch case (learned in C when coding for unity)
    print ends her turn statement
    else return nothing

int main(void)
    int numplayers = promptNumberofPlayers()
    unsigned seed = promptSeed()
    prng(seed)
    int playerchips[numplayers]
    startingChips(playerchips, numplayers)
    while checkWinner(playerchips, numplayers)== -1)
        for int i = 0; i < numplayers; ++i)
            if checkWinner(playerchips, numplayers)== -1)
                turn(playerschip, i, numplayers)
    int winner = checkWinner
    printf(%s won!\n, player_name[winner])
    return 0
```

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge.

References