Assignment 2 – LRC Report Template

Mira Saini

CSE 13S – Spring 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

This program mirrors a dice game called "Left, Center, Right." The program first takes a user input indicating how many players are in the game. Each player is initially assigned 3 chips. The program checks the number of chips before the player's turn to assess how many dice the player should roll. The first player enters a random seed value that generates a number between 0 and 99,999. The program takes this number and uses the modulus operator to divide by 5, which returns a remainder value between 0 and 5 inclusive. The program repeats this process for however many dice the player would roll in the original version. The values 0, 1, and 2 are assigned to "DOT," which in the game, means that the player does not give any chips away. Value 3 is assigned to position left, 4 is assigned center, and 5 is assigned right. Depending on what the player rolls, chips are given to the player on the left, to the center pot, or to the player on the right. Players with no dice are skipped over. The program uses a function to update the number of chips each player holds. The program then assesses, using a while loop, if at least twice players have dice in the game, otherwise the game is over and a winner (whoever has leftover dice) is declared.

1 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.

1.1 Randomness

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

Randomness should not have a recognizable pattern and the situation/current outcome of a game, in this case, should not have an impact on what number is generated. It's not really possible for anything to be truly random since random algorithms are created by pattern-making humans. In a random generator with a wide-range of numbers, there's a greater sense of randomness because repetition is not as common. In games like rock, paper, scissors, even against a bot with a random algorithm, patterns may emerge early-on until the game is repeated a lot of times, where the chance of each decision - "rock," "paper," or "scissors" - comes out to be equally likely (around 33.33 percent). We are using pseudo random numbers because the built in random number generator in C's standard library will do different things based on one's operating system. To standardize this, the pseudo random number generator provided to us lets the grading computer and our computers have the same sequence of generated numbers.

1.2 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 simplified sounds like "abstract" meaning without very much detail. In non-computer science terms, an abstraction encompasses the general idea of a subject, but isn't too focused on the specifics. It's more about the idea rather than the execution of it.

1.3 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?

I think abstractions can help to get the general idea of what it is you want to do with a piece of code. Instead of focusing on super specific details, one can abstract their code to find out what their program is really intending to do in order to remove any unnecessary details that might make the program buggy or irrelevant. Abstractions can also be used to outline your code before you begin actually writing it so you can get a general idea of how to structure your program. I don't think you have to be the one to write an abstraction because a general outline as to how to write the program can be used by anyone; then as the developer, the program can become more specific based on your preferences.

UPDATE: I definitely was using a lot of abstraction techniques to decipher and debug my code. I didn't realize a few small bugs that completely hindered my program output until I put in a bunch of print statements in my program to check the switch of chips between players for every dice they rolled. This helped me pin point the two changes I needed to make (thank goodness). I really did break my searching up by focusing on the functions separately, which i'm so glad I included functions otherwise this would have been much harder to debug.

1.4 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? 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.

We can breakdown the steps of the program into separate functions to make the code more simpler to read. If we had to write the program with 2 functions + main, I would designate a function to taking in user inputs and another function to update the amount of chips each player has every round. If the program used 8 functions, these steps could be broken down even more. Two separate functions could be used to print the parameter of the name of the player or the number of chips they have. A function could also be used to determine how many dice a player should roll which inside that function, the update_chips function is called with the parameter being the number of dice that the player receives. These functions can be further program down to create more than 8. Contrasting an implementation with function versus an implementation with no functions, functions make code easier to debug and read and provide more of a flow to the program. If you didn't use any functions, there would be more lines of repeated code, where only one variable may be altered. I also think that many more if statements would be involved in a program with no functions.

UPDATE: I decided to create an extra function in my program in addition to writing the update_chips function. The to_exit function takes in the chips array and loops through it looking for values that are more than 0 after the player has gone through their turn. If the function finds a value that is more than 0, it add it to a counter variable. Then the counter variable is checked to make sure that it is at least 2, meaning that at least 2 players have chips. If the counter variable doesn't find at least two players with chips, it returns the name of the player that has won. Then the function returns back to the while loop and then a string comparison is done to check what the function returned (either "bad" if the game should continue or the winner if the game ended) comparing it to "bad". The comparison returned either a 1 or 0 which then was checked (1 being that the strings were different, 0 being the same). If the comparison function returned a 0, then the game continued otherwise the game ended.

1.5 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.

To make my tests comprehensive, I can test inputs that are boundary numbers and that incorporate both type I and type 2 tests.

For my Type I tests (inputs that are correct), I could test to make sure that the program correctly accepts the number of players = 11 and the seed number = 99,999 to test the high boundary of both variables. To test the lower boundary, I would use inputs 3 and 0, expecting these inputs to work. For Type 2 test (inputs are erroneous), I would test the other side of the boundary, making sure that program does not accept combined inputs like 12 and 99,999, 11 and 100,000, 3 and 100,000, and 2 and 99,999, where at least one input is erroneous (too big or too small). I can also test to make sure that inputs are not symbols, letters, or non-positive non-integers (-3.5) that do not fall in the ranges as well.

1.6 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? A pseudo random number generator standardizes the sequence of generated values on both the grading and student computers. Instead of wondering why a program works for one laptop and not another with a different OS, I won't have to ensure that problem arises because the pseudo random number generator prevents that from happening for me. Additionally, writing abstractions for each part of the program will break down the program into sizeable bits that are easier and more straightforward to code. Debugging code that is sliced into separate functions will also be easier as the code is neatly designated to functions that perform separate tasks.

UPDATE: Now I understand a bit more about the pseudorandom generator that was provided to us. It really helped that I could replicate my sequence of numbers with a random seed and how the lcr_x86 replicated the same sequence if it was the same seed. It made checking my program output so much easier and it was less of a hassle to make sure that I was doing the correct things instead of constantly running my program through the pipeline.

2 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 use this program, simply enter the amount of players in the game when prompted. The program will also ask for a seed number for each player per round which will generate and mimic a random dice roll. The program will then designate the chips according to the outcome of the dice roll(s) and then you will be prompted to repeat this process until the game ends, where only one player has chips. To run and compile the program, simple type the keywords "make" into the terminal to compile and "./lcr" to run the program.

UPDATE: Yeah, there is no update for that part really. It's still the same.

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]

[1] This is my footnote.

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 this[1][2][3].

3 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.

This program is organized in the following way: while loop (checks at least 2 players have more than 0 chips) for loop (loop through players): this is where the program indicates through a print statement who is rolling the dice User enters seed number, error catch is written here determine how many dice player should roll, dice = num of chips contains my update_chip function which uses a for loop to repeat the process n amount of times and then prints out the player and their chip value at end of turn Outside of the while loop: prints the player who won

Some of my main data structures are hash tables/dictionaries. I will be using this to store a key value pair of player to chips (this is how I will keep track of the chips that each player has). In the beginning of the program, all values in the hash table will be set to 3. The player names uses an array, so I will also be indexing from the array using the first for loop to go through all of the players.

UPDATE: So I ditched the for loop and basically just used a while loop. I then had my dice roll simulate at the beginning of the loop, which then called the update_chips function with the number of rolls as a parameter. Then I made sure to update my chips in that function, return to main, and then call the to_exit function which checked if at least two players had chips. The function returned and did a string comparison of the returned value and the string "bad" and if they were the same, the game continued otherwise the game exited and a winner was declared.

3.1 Overall Pseudocode

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

Note: I might be using a hash table/dictionary to keep track of the number of chips each player has so my pseudocode indexes a hash table

```
#While loop to make sure that at least two players have more than 0 dice
#For loop, to loop through the player names
#print the player rolling the dice: printf(%s is currently rolling the dice, player[i]);
#User enters appropriate seed number, error catches if seed number invalid (code already given in pdf)
Determine if and how many die the player should roll:
if player_name_chips = 1:
#player rolls one dice
# call the update_chips function(1)
elif player_name_chips = 2
#player rolls two dice
# call the update_chips function(2)
elif player_name_chips ¡= 3
#player rolls 3 dice
# call the update_chips function(3)
else: #player has 0 dice, players will never have negative dice counts
#player rolls 0 dice, skips turn
#update_chips function (parameter = number of rolls)
#For i in range(parameter of function):
To get the type of roll, must do cse_random() % 5 which will give numbers 0-4 equation to positions (dot, center, left, right)
```

Bunch of if conditions:
if roll == dot:
#do nothing
elif roll == center:
#chip gets placed into pot
elif roll == left:
#chip placed into player_num-1 if not player zero, if player zero then amount of players + 0
elif roll == right:
#chip placed into player_num+1 if not max_num player, if player is max_num then amount given to player 0
#this is in the update function btw since players with 0 chips are skipped and their number of chips is not printed either
printf("%s ends her turn with %d\n chips", player_name, player_num_chips)
#When the while loop exits, this means that only 1 player has dice, ending the game
printf("%s won!\n", winner")
UPDATE: Basically, almost everything is the same besides the added to_exit function and the deletion of the unnecessary for loop.

3.2 Descriptions of Functions

For each function in your program, describe 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.

I only will be using one function in my program to update the number of chips each player has. The inputs of the function will be the number of dice the player is going to roll. So if the player is rolling one dice, the function call would be update_chips(1). There are multiple outputs of the function (not a return value), depending on what the dice roll lands on. The will be a for loop to repeat this process for as many dice the player is going to roll. If the outcome of the roll == dot, then the program will not shift any chips, roll == center then a chip gets placed into the current pot (add to some variable), if roll == left, then chip value is added into player_num- 1 if not player zero, but if this is player zero then the chip is given to player number equal to amount of players + 0. If the chip == right, then the chip is given to current player_num + 1 as long as the max_num player (player assigned to go last), if player is the last to go then the amount is given to player 0. Then the function will return a print statement stating that the current player ends her turn with a specific number of chips. Note that the print statement is not in the for loop because this statement should only be printed once after the player's full turn.

Here is some pseudocode: for i in range(parameter of function = number of dice to roll):

To get the type of roll, must do cse random() % 5 which will give numbers 0-4 equation to positions (dot,center, left, right)

Bunch of if conditions to find outcome of roll:
if roll == dot:
do nothing
elif roll == center:
 chip gets placed into pot
elif roll == left:

chip placed into player num-1 if not player zero, if player zero then amount of players + 0

elif roll == right:

#chip placed into player num+1 if not max num player, if player is max num then amount given to player 0

#The print statement below is in the update function since players with 0 chips are skipped and their number of chips is not printed either, but this statement is outside of the for loop

return printf("%s ends her turn with %d\n chips", player name, player num chips

UPDATE: Description of the to_exit function: this function basically loops through the chips arrays and checks that each element is greater than 0. If it is, then it adds 1 to a counter variable. Then the counter variable is checked to make sure that it is greater than 1. If so, the program returns "bad". If not, the program returns the name of the winner. Once the function returns to main, I wrote a string comparison part that compared the returned part of to_exit with the string "bad." If the strings were the same, that meant that the exit code that would be returned would be a non-zero. So I wrote an if statement that said if comparison returns 0, then the game continued but if it was non-zero the game ended.

Boom. It's currently 10:15pm of the night that this assignment is due and I'm....done.

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.