Winston To

Prof. Jeff White

IT 312

1/24/2021

Final Project Documentation

2/21/2021

Final Project Release 1.0 & reflection:

Code has been written, the game is fully functional and working. the overall hierarchical structure of source code really resembled my pseudocode, minor changes were made in handling the values and variables for the player objects. I didn’t feel comfortable passing around an array of objects into functions so the majority of the roll resolve happened in the RunGame() function of the Game Class. Formatting was done to match with the text output/read. Keeping track of all the pointers was perhaps the most challenging part of this project and making sure that I am referencing the proper value for each player. There were several errors made regarding syntax. While developing the code to resolve the roll results, I realized I had to add in code to make sure the player to the left/right was within the index range of the players available. The first player object in the array had to pass to the last player if the result was L, and the last player had to pass to the first if the result was R. Calling functions from classes, I sometimes tried to set class variables with the = operand instead of using the setters/getters. These were simple fixes. I am very happy with the solution; however, it seems the random generator seems to not be truly random as the results for every game seem the same. Not quite sure how to fix it yet; I will re-visit this. I have employed Test driven development throughout the project, building every time I implement a new function and outputting variables to ensure I am properly referencing the desired value.

I have incorporated an endgame special feature that outputs the following:

A round counter, chips in pot count, and winner stats (chips).

2/14/2021

I have written the rules.txt of LCR and formatted them for output. I intend on using this work from this week’s assignment in my final project. I have also reflected on the usage of dynamic arrays, something I missed in my studies. They are something I will incorporate into my code along with classes due to the nature of the game having a non-static player count. Everything I have learned so far has opened new ideas and approaches to the developing my game.

2/6/2021

I have further revised my draft pseudocode. It is still very rough but when I learn how to better handle header files and classes, along with OO programming I intend on revising it further. After completing the fill in the blank assignment for this week’s assignment I realized I am going to have to practice some more on writing classes and organizing the proper function hierarchy for a game. I may model this dice game after that assignment. It had a solid loop function and game option menu.

1/31/2021

After completing and implementing the rand() function for the dice roll assignment, I realize that it will be a major part of the final project. I will utilize the function I created to make the rolling mechanic for the 3 dices in the game. Coming up with a mechanic to calculate the outcomes and process what to do next will be the next part of my brainstorming process. I need brush up on classes and object-oriented concepts in C++. I will utilize loops, if conditions, and Boolean variables to process the outcome of the dice rolls. I have furthered outlined the requirements from the final rubric and will begin writing a draft pseudocode.

1/24/2021

I have outlined the requirements for the game as well as the rules of the game itself (as shown on the last page). My next step involves reviewing the rubric and finding out what coding requirements the final project will require. I plan on outlining that as well. From there I will begin writing pseudocode and determine if any more preparation will be required, such as reviewing material or learning new C++ operations. Once my outlines are complete, I will write commented pseudocode for the game in the IDE as well as begin on the source code. Upon writing the source code, I will perform some TTD, test driven development where I will test each function to make sure that my code and logic is sound.

1/17/2021

This document will outline the requirements, as well as track and document the progress of my final project. The game I have chosen to make is LCR, a dice game using three special dice and player pieces called chips. I have so far reviewed rules of LCR and am in progress of analyzing the problems I may face. As soon as I outline requirements of the rubric and game, I will begin writing the pseudocode for it. Perhaps create class diagrams for the object-oriented portion if necessary.

**Game Requirements**

-Output rules from text file on program start-up

-Prompt user to set up the game by entering the number of players

-Game requires a minimum of 3 players, ask user to add more players if below 3.

-Dice Roll Mechanic

**Game Rules**

-Each player starts with 3 chips

-Roll 1 is L, pass one chip to player sitting to the left

-Roll 2 is R, pass one chip to player sitting to the right

-Roll 3 is C, chip goes into center pot and are now out of play

-Roll 4-6 can be represented with \*

-Check current player’s number of chips before rolling.

1 chip = 1 die

2 chip = 2 die   
3 chip = 3 die.

-After each player’s roll, calculate the number of chips for players based on the actions dictated by the dice. Continue playing until only one player has chips

-Display a message to the game winner.

-Number of Dice rolls == to number of chips player possesses

**Technical requirements.**

1. Input/Output: Your program reads input from a file and uses system input and output through the console.

B. Control Structures: Your program utilizes appropriate control structures for game logic.

C. Libraries: Your program uses standard libraries to pull in pre-defined functionality.

D. Arrays and String Manipulation: Your program effectively uses arrays and string manipulation.

E. Classes Breakdown: Your program is broken down into at least two appropriate classes.

F. Methods: Your program uses all included methods correctly within the classes.

G. Error Free: Your program has been debugged to minimize errors in the final product. Your instructor will run your program to determine functionality.

H. Error Documentation: Accurately document major errors that you encountered while developing your program.

I. Solution Documentation: Document how you solved the errors and what you learned from them.

J. Formatting Best Practices: Your program code is easy to read, following formatting best practices as defined by the industry, such as with indentation.

K. Documentation Best Practices: Your program contains comments where needed, in appropriate detail for communicating purpose, function, and necessary information to IT professionals.

L. Coding Best Practices: Your program supports clean code through descriptive variable names.

**Psudocode**

**Load rules.txt**

**Output txt file**

**Game Starts**

**Prompt user to enter number of player (requires min 3 players)**

**Create array of player classes**

**set endConditionSkips = length off array - 1**

**Run Game**

**int skips = 0**

**Loop until skips == endConditionSkipps**

**skips = 0**

**If Player has chips**

**Player Goes and rolls Dice == Chips in their possession**

**Rolls get resolved**

**Else if player does not have chips**

**Player skipps**

**skips add 1**

**Next Player Goes**

**Game Ends**

**//Player Class**

**//Variables**

**Int playerChips = 3**

**int Array playerDices[3] = 0**

**bool hasChips = true**

**//Player roll Function**

**if player hasChips**

**loop for i = 0 to i == hasChips i ++**

**randomly generate 1-6 and store in player dice[i]**

**roll resolve**

**else if player !hasChips**

**skipp roll**

**//Roll Resolve function**

**caseSwitch**

**If player rolled a 1 in their turn, pass chip to player left of them**

**If player rolled a 2 in their turn, pass chip to player right of them**

**If player rolled a 3 in their turn, chip goes out of play into center**

**If player rolled 4 – 6, no action is taken**