Project 2

Yahtzee Game

CSC-5-42521

Martin Schmid

June 1, 2019

# Rules of the Game

For Project 2, I enhanced my partial­­ implementation of the dice game *Yahtzee*. The goal of the game is to achieve the highest possible score. The game is comprised of six rounds. Each round, five dice are rolled, and their values are displayed to the player. The player must decide each round what value they would like to score, from 1-6. The goal is to have as many dice as possible at that round’s value, maximizing the score each round. There are two dice rolls: an initial roll of all five dice, and an optional second roll of any number of dice selected by the player.

For instance, a player’s initial roll could result in dice values of 3, 1, 3, 4, and 3. In this situation, a smart player will aim to score threes for the round, so they will reroll the 1 die and the 4 die. Now their result is 3, 3, 3, 4, and 3. They can score threes that round and get 12 points (3x4), which is a very good score for a round of threes.

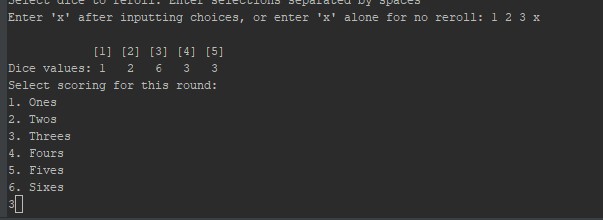
Once the player has scored for the round, any points earned are added to their total for the game, and they proceed to the next round. Each number 1-6 can only be scored once. The player that earned 12 points in the round of threes can no longer score with threes and must make different selections for each subsequent round. At the end of six rounds, the points are tallied up.

The game *Yahtzee* traditionally includes “poker” scoring options such as Full House, Straights, 3- or 4-of-a-kind, etc. My simplified version does not include these scoring options.

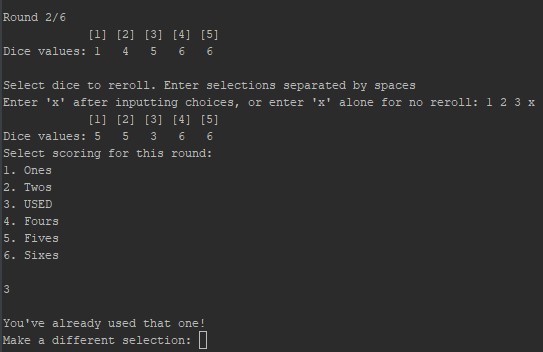
# The program in action

# 

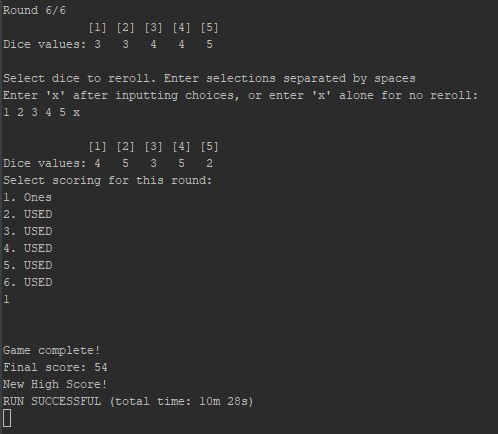
In the above screenshot, we can see the first round of a game. Dice 1 and 2 resulted in one, die 3 resulted in two, and dice 4 and 5 resulted in three. The player must input the dice numbers (if any) they’d like to reroll, followed by the letter ‘x’ to mark the end of their selection. In this example, the player is rerolling dice 1-3, trying to get as many threes as possible this round.



Unfortunately, the second roll did not add any more threes for the player to score. But as we can see at the bottom, they are still choosing threes to score for the round, because there are more threes than any other number. The player’s score for this round will be 6.



The above screenshot shows the next round in action. One thing to note: since the player scored threes the previous round, that option shows up as “USED.” The player cannot score threes for more than one round. This is further reinforced when the player attempts to select threes anyway, and receives an error message.



The above screenshot shows the final round of the game. After the scores for each round are tallied, the final score is 54. The program compares this to the existing high score file, and overwrites the existing score if the new one is higher.

A close up of a map

Description automatically generated

Cross-List\_From\_Proj1

**Cross Reference from Project 1**

**You are to fill-in with where located in code**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 2 | 2 | cout | 31 |  |  |
|  | 3 | libraries |  | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime |
|  | 4 | variables/literals | 27 |  | No variables in global area, failed project! |
|  | 5 | Identifiers |  |  |  |
|  | 6 | Integers | 27 | 1 |  |
|  | 7 | Characters |  | 1 |  |
|  | 8 | Strings | 29 | 1 |  |
|  | 9 | Floats No Doubles |  | 1 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 28 | 1 |  |
|  | 11 | Sizeof \*\*\*\*\* |  |  |  |
|  | 12 | Variables 7 characters or less |  |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables |  |  |  |
|  | 14 | Arithmetic operators |  |  |  |
|  | 15 | Comments 20%+ | throughout | 2 | Model as pseudo code |
|  | 16 | Named Constants |  |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate |  |  | Emulate style in book/in class repositiory |
|  |  |  |  |  |  |
| 3 | 1 | cin |  |  |  |
|  | 2 | Math Expression |  |  |  |
|  | 3 | Mixing data types \*\*\*\* |  |  |  |
|  | 4 | Overflow/Underflow \*\*\*\* |  |  |  |
|  | 5 | Type Casting |  | 1 |  |
|  | 6 | Multiple assignment \*\*\*\*\* |  |  |  |
|  | 7 | Formatting output | 183 | 1 |  |
|  | 8 | Strings | 29 | 1 |  |
|  | 9 | Math Library |  | 1 | All libraries included have to be used |
|  | 10 | Hand tracing \*\*\*\*\*\* |  |  |  |
|  |  |  |  |  |  |
| 4 | 1 | Relational Operators |  |  |  |
|  | 2 | if | 246 | 1 | Independent if |
|  | 4 | If-else | 43-45 | 1 |  |
|  | 5 | Nesting | 63-145 | 1 |  |
|  | 6 | If-else-if |  | 1 |  |
|  | 7 | Flags \*\*\*\*\* |  |  |  |
|  | 8 | Logical operators | 67 | 1 |  |
|  | 11 | Validating user input | 76, 89, 102, 115, 128, 141 | 1 |  |
|  | 13 | Conditional Operator | 59 | 1 |  |
|  | 14 | Switch | 65 | 1 |  |
|  |  |  |  |  |  |
| 5 | 1 | Increment/Decrement | 68 | 1 |  |
|  | 2 | While | 145 | 1 |  |
|  | 5 | Do-while | 63-145 | 1 |  |
|  | 6 | For loop | 177 | 1 |  |
|  | 11 | Files input/output both | 155-65 | 2 |  |
|  | 12 | No breaks in loops \*\*\*\*\*\* |  |  | Failed Project if included |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not | required to | show | Total | 30 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 6 |  | Functions |  |  |  |
|  | 3 | Function Prototypes | 15 | 4 | Always use prototypes |
|  | 5 | Pass by Value |  | 4 |  |
|  | 8 | return | 172 | 4 | A value from a function |
|  | 9 | returning boolean |  | 4 |  |
|  | 10 | Global Variables |  | XXX | Do not use global variables -100 pts |
|  | 11 | static variables |  | 4 |  |
|  | 12 | defaulted arguments |  | 4 |  |
|  | 13 | pass by reference |  | 4 |  |
|  | 14 | overloading |  | 5 |  |
|  | 15 | exit() function |  | 4 |  |
| 7 |  | Arrays |  |  |  |
|  | 1 to 6 | Single Dimensioned Arrays | 27 | 3 |  |
|  | 7 | Parallel Arrays | 209-213 | 2 |  |
|  | 8 | Single Dimensioned as Function Arguments |  | 2 |  |
|  | 9 | 2 Dimensioned Arrays |  | 2 | Emulate style in book/in class repositiory |
|  | 12 | STL Vectors | 192 | 2 |  |
|  |  | Passing Arrays to and from Functions | 197 | 5 |  |
|  |  | Passing Vectors to and from Functions | 197 | 5 |  |
|  |  |  |  |  |  |
| 8 |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort | 241 | 4 |  |
|  | 3 | Selection Sort | 223 | 4 |  |
|  | 1 | Linear or Binary Search | 68-69 | 4 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not | required to | show | Total | 70 | Other 30 points from Proj 1 first sheet tab |