Program #5

Due Nov 18 by 11:59pm **Points** 100

Implementing Yahtzee

Introduction

For this homework project you will have two weeks to complete the programming portion of the assignment. **However, you will be required to submit a design document to Canvas (due on Sunday, November 11th, 11:59pm).** You can use the same documentation template and format that was introduced during week 2. Please review the **recitation syllabus (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/CS161_Recitation_Syllabus.pdf)** (available from the Recitations tab) for additional details on the design document expectations.

Example design document (introduced in week 2): **Polya_template.pdf (https://web.engr.oregonstate.edu/~goinsj/resources/CS161/Polya_template.pdf)**

This particular program implements a game that is played in real life. For this assignment you should try to make your program look as nice as possible.

(85 points) Problem Statement

Write a C++ program that plays the game of Yahtzee. Here are instructions for playing the game: **http://www.hasbro.com/common/instruct/Yahtzee.pdf (http://www.hasbro.com/common/instruct/Yahtzee.pdf)**
You can also find online implementations if you want to see the game in action. One specific online implementation is at: **https://cardgames.io/yahtzee/ (https://cardgames.io/yahtzee/)**

Your game will allow 1-2 players, and at the end, you need to ask if the users want to play again. There are 5 six-sided dice in the game and you get three rolls to try to make something on the scoresheet. You must use an array to hold the values of the 5 dice and an array to hold the scores for the 16 categories for a player.

The top half of the scoresheet is only a sum of 1 or more of dice you are trying to get: 1s, 2s, 3s, 4s, 5s, and 6s, and if you score 63 or more points in this section, you get an additional 35 points.

The bottom half of the scoresheet consist of options that have pre-defined points or a total of all the dice. You have the options for a three of a kind (total of all dice, but has to have 3 or more of the same number), four of a kind (total of all dice, but has to have 4 or more of the same number), full house (25 pts, 2 dice the same and 3 dice the same), small straight (30 pts, 4 numbers in a row), large straight (40 pts, 5 numbers in

a row), Yahtzee (50 pts, 5 dice the same number), up to 3 bonus Yahtzees (100 pts each, 5 dice the same number), and chance (total of all dice, any combination).

After your first roll, a player can choose to roll 0-5 specific dice again. The player needs to state how many dice they want to re-roll and which ones they want to re-roll by providing the array indices for each die. If the player chooses not to roll again or has rolled 3 times, then the player must choose a category not previously chosen on the scoresheet. If the dice do not match the rules for that category, then the player takes a 0 for that category. Once a category has a score (even a zero score), it cannot be chosen again, so think carefully about what value you are going to use to initialize the scoresheet!!!

The bonus Yahtzee can only be chosen if there is a non-zero score in the Yahtzee!!! The player's turn is over once a category is chosen, and the game is over once all players have put a score in all categories. The final score is a total points from the two sections, including the bonus if the top section is 63 or greater.

Command Line Arguments

To support various numbers of players, you will run the program with the specific number at runtime, rather than prompting for the input. The user can enter the number of players, 1-2.

1-player Example:

```
./yahtzee 1
```

Additional Requirements

In addition to the earlier specifications, your program must meet these requirements:

- Print an error message and recover, when the player doesn't supply a valid category. This includes selecting a category with a score, which can include a zero for the score. Make sure to carefully consider this when you are designing your program!
- Update: as announced in class on Friday, you can safely assume that the TAs will only test your program with the command line values of "1" and "2" (unless you also did the extra credit). So you no longer have to halt your program if other input is provided.
- Correctly determine the score for the category based on the dice (Remember, if the dice do not match the rules for the category, then a zero score is placed in that category).
- Play the game correctly based on rules and number of players.
- Continue to play until the user selects no.
- You must not have any global variables
- You must use a dynamic 1-d array for the second player's scoresheet, since there may or may not be a second player.
- Your functions need to focus on performing a particular task. In other words, you need to use good modular design. If your function uses more than about 20 lines of code, this may be an indication that the code should be split into multiple functions. If the TA notices that your code is not sufficiently modular, you will lose points.

- You must not have memory leaks.
- Segmentation faults are not allowed (e.g. your program crashes).

(10 pts) Extra Credit - Supporting any number of players

Instead of having only 1-2 players, your Yahtzee will support any number of players using a dynamic 2-d array. This would have 16 rows/categories by N players.

(15 pts) Program Style/Comments

In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the **style guidelines for this class (https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)** , and try to follow them, i.e. don't align everything on the left or put everything on one line!

```
/*************************************************
** Program: yahtzee.cpp
** Author: Your Name
** Date: 11/7/2018
** Description:
** Input:
** Output:
*************************************************/
```

Don't forget that each of your functions needs to have a header. For example:

```
/*************************************************
** Function:
** Description:
** Parameters:
** Pre-Conditions:
** Post-Conditions:
*************************************************/
```

Assignment Submission

Note that there are two deadlines for this assignment.

Your design document must be scanned and electronically submitted to Canvas (in the assignment tab) by **Sunday, November 11th, 11:59pm**. You may use one of the document scanners (available in KEC1130 and some other College of Engineering labs) to scan your paper design into a PDF document that can be submitted to Canvas.

Electronically submit your C++ source code to **TEACH (https://engineering.oregonstate.edu/teach)** by the assignment due date on **Sunday, November 18th, 11:59pm**. Your TEACH submission should contain the .cpp file with your source code.

Reminder

Every assignment in this course is graded by demoing your work for 10-15 minutes with a TA. You are required to **meet with a TA within two weeks of the due date** to demo. You can schedule a demo with a TA from the **TA Office Hours** tab in Canvas. The available times can be viewed in the far right column of the table (labeled "Grading Hours"). Click on one of the links to access the poll and insert your name in a specific time slot.

- **Demo Outside 2 Weeks**: Assignments that are not demo'd within the acceptable time period will be subject to a 50 point deduction.
- **Demo Late Assignments**: Late assignments must still be demoed within the two week demo period beginning from the assignment's original due date.
- **Missing a Demo**: If you miss your demo with a TA, you will receive a 10 point (one letter grade) deduction to that assignment for each demo missed.

Each program needs to be written according to the **style guidelines (https://web.engr.oregonstate.edu/~goinsj/resources/general/cpp_style_guideline.pdf)** for this class. Remember that an important part of computer programming is making sure that your work is easily understandable by other programmers.

Program #5 Rubric

Criteria	Ratings	Pts
Program Header At a minimum, header should contain author's name and a description of the program. (2 pts each) 1 point for submitting code that compiles on flip.		5.0 pts
Good indentation / Use of whitespace Is code easy for the TA to read? Conditional blocks of code should always be indented.		5.0 pts
Each function is documented Every function contains it's own initial block comment (or multiple lines of comments prior to the function definition) that provides the reader with an explanation of the function's purpose. -1 pt for each function that is missing a header		5.0 pts
All functions designed in a modular fashion If functions are exceptionally long (greater than about 20 lines of actual code) this is a potential indication of poor modularity. -3 pts for each function that the TA concludes is poorly modularized.		12.0 pts
Program can run with 1 player or 2 players via command line argument.		5.0 pts
Dice Operation Dice values are randomized after a roll (3 pts). Player can only roll dice up to three times (3 pts). An array is used to store the 5 dice values (3 pts). The user can choose to keep certain dice and those values do not change with a re-roll (3 pts).		12.0 pts
Category Validation Program needs to print an error message and recover if the player tries to supply an invalid category. Examples: Trying to use the same category after it's been used (even if the score was 0) (5 pts). Trying to claim points for a non-existent category (e.g. pick category 72) (5 pts).		10.0 pts
Array Usage Scores are stored in a array (6 pts). Student used a dynamically allocated array (i.e. must use "new" operator) for the second player (6 pts).		12.0 pts
Program Functionality Program does not crash during testing (e.g. segmentation fault or similar abrupt halts) (8 pts). User can choose to replay the game (2 pts). No global variables are used in the program (2 pts).		12.0 pts
Display of Categories Game displays 16 categories that can be scored. It is okay if the 3 bonus Yahtzees are hidden from the player (if there is no Yahtzee, the user is not allowed to choose a bonus Yahtzee). If the score is not computed until the user selects the category, that is still okay too.		5.0 pts
Category Scoring Game correctly computes the value of each category. E.g. Score is correct for each of the 16 categories. -3 pts for each category that computes incorrect score. Specifically test: 3's, 6's and Chance. If other possibilities emerge during play, test those too.		9.0 pts

Criteria	Ratings	Pts
Memory Management Running Valgrind on the program reports that "All heap blocks were freed" (or something indicating that all memory is freed). There should be no blocks reported as "in use" at exit or "possibly lost".		8.0 pts
Extra Credit for additional players (10 pts) Support additional players by implementing a dynamic 2D array for the scores. Game can support at least 4 players.		0.0 pts
		Total Points: 100.0