Assignment 5
Due: Sunday, 4 March 2018, 11:59 PM on TEACH as a .cpp
Farkle

Background
Farkle is a dice game designed for two or more players. Gameplay consists of throwing six dice and gaining points based on single rolls or special combinations of dice rolls described below. When the first player reaches 10,000 points, each other player gets one more turn to attempt to beat the first player's score. After this special set of turns is taken, the player with the top score wins.

In order to get on the scoreboard, you must have a running total of at least 500 points before you stop rolling. Once a player is on the scoreboard, they may choose to stop rolling at any time during their turn. Once a player chooses to finish a sequence of rolls, their score is added to the scoreboard and their turn ends.

Each turn starts by rolling all six dice. After each roll, set aside the dice that are worth points and roll the rest. You must remove at least one die or combination of dice that are worth points after each roll while keeping a running total of your points for the turn. If you are able to set aside all six dice, you can roll the dice again to start the process over and build up additional points or you can keep the points you have and end your turn. If you can not set aside any dice after a roll, this is called a "Farkle". When a Farkle occurs, points are not added to the scoreboard and play passes to the next player. A Farkle may happen on the first roll or any subsequent rolls.

Here is an example of the game: https://www.youtube.com/watch?v=zpdHZ8AUAtQ

(Instructions continue on the next page)

Scoring

Single 1 = 100	Four of any number = 1,000
Single 5 = 50	Five of any number = 2,000
Three 1's = 300	Six of any number = 3,000
Three 2's = 200	1-6 straight = 1,500
Three 3's = 300	Three pairs = 1,500
Three 4's = 400	Four of any number with a pair = 1,500
Three 5's = 500	Two triplets = 2,500
Three 6's = 600	

Implementation Requirements

· Must produce a working program that implements the game play described in the Background and Scoring section of this document.
· Rand() should be used to simulate the dice rolls.
· All functions should be 15 lines or less. Whitespace, single curly braces, and the function header do not count.
· Use of one dimensional arrays is required.
· Use of references and pointers is required.
· The user should see their running score total for their turn at all times.
· The user should be able to choose which dice to remove from the rolls during their turn.
· Error handling for all user input should be implemented.
· The scoreboard for all players should be displayed at the end of each turn.
· The program should always state whose turn it is.
· The program should allow as many players as the user request, but there should be a minimum of two.
· No memory leaks allowed (see valgrind).
· No global variables allowed.
· No segmentation faults allowed.

(Instructions continue on the next page)

Grading

Design to be submitted to recitation.
Implementation: 80 pts
Auto deductions:
Memory Leak: -10 pts
Global Variables: -10 pts
Segmentation Faults: -10 pts
Style and Code Quality: 20 pts
Extra Credit: 10 pts

Extra Credit

Log all testing activity that you conduct on your program. You should produce a table as follows:

Date and Time	Function or Procedure Being Tested	Preconditions to get to testing scenario	Expected Post Conditions	Actual Post Conditions	Hypothesis on the cause of the issue	How the issue was fixed
…	…	…	…	…	…	…

Note, this is not the same as the testing table you normally produce for your recitation designs. This is a log of all the issues you've encountered in developing your program and how you fixed these issues. By recording what goes wrong, you should be able to better recognize common mistakes and identify how to fix them quickly. This table should span multiple pages for this program.