# Program Design & Testing Document for Program 5
# Lyell C Read

**Problem Statement**

The problem asks me to create a c++ version of the Yahtzee game. This game must:

- Supports n players, and this choice must be made when the executable is executed, in the command line.
- 5 dice roll 3 times with the ability to keep dice.
- Dice and Scoresheet stored in an array
- Choice not to roll again
- 35 point bonus for >63 in top part of scoreboard

More requirements (credit: Justin Goins, Canvas):

- 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!
- Print an error message when the user enters an invalid option as a command line argument. You do not need to recover from this.
- 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).

**Understanding the Problem**

As described in the requirements, the problem asks me to create a game of yahtzee with the above listed constraints. This game will start with the user calling the executable with a parameter on 1..<infinity> of how many players will play the game.

**Pseudo Code (Simplified)**

//when I write this for real i will make a dedicated function to create the deltas array and return the pointer to it.

```
q_of_n (n, &dice):
        For every instance of n in array of dice, q++
        Return n

n_of_a_kind_check (n, &dice) // n = 3 or 4 of a kind
        For 1..6 = num, call q_of_n:
                If q_of_n >= n :
                        Return sum (dice)
        Return 0

Full_house (&dice)
        sort dice in increasing order
        if ((q_of_n(dice[0]) == 2 && q_of_n(dice[2]) == 3) || (q_of_n(dice[0]) == 3 &&
q_of_n(dice[3]) == 2)
                Return 25
        else
                Return 0

sm_straight  (&dice)
        Sort dice in increasing order
        New array deltas
        For every item in dice 1,4:
                deltas[item] = dice[item]-[dice[item-1]
        //[1],[2],[3],[4],[6] --> deltas: [1],[1],[1],[2]

        Either hard check deltas for 1,1,1,*;  *,1,1,1 or hard check list (ugly)
        If those patterns are present
                Return 30

        Return 0

lg_straight (&dice)
        Sort dice in increasing order
        Create deltas array
```

```
        If q_of_n(1, deltas) == 4
                Return 40
        Return 0

Sum_of_array (array)//if this is not part of some built in fxn
        For x in array
                Return += array [value]
        Return return

Y_check (&dice)
        Create deltas list
        If q_of_n (0, delta)
                Reutrn 40
        Return 0

Numbers (n, &dice)
        For x in array
                If x == n
                        Return_val += n
        Return return_val

dice_roll(&dice)
        Generate random for each value in dice
        Do twice:
                Ask for keepers (either binary 10010 or one by one, maybe GUI?)
                For each one of dice if it is supposed to be randomized,
                        Generate random for that cell, replace
        Print the final dice arrangement

main()
        Seed random
        Generate the scoresheet array as a 16 X arg[0] which is one “column” for each player.
        Print welcome message
        Do 16 times:
                For each player
                        New array for dice
                        Do dice_roll
                        Print out the current scoreboard with the relative possible scores
                        Ask where to play dice
                        <sequence of if else here - these call the respective functions to evaluate
points, and score correctly in the score sheet array>
        Evaluate who won, and print final grid.
        exit()
```

**Predicted Results for Only Integer Input**

| Value | What Should Happen | Does This Happen |
|---|---|---|
| “” | Error - nothing entered | yes |
| “2” | 2 | yes |
| “oasdjfasone” | Error - improper formatting | yes |
| “;” | Error - improper formatting | yes |
| “ “ | Error - improper formatting | yes |

**Predicted Results for Execution**

| Value | What Should Happen | Does This Happen |
|---|---|---|
| ./a.out 2 | 2 player game | yes |
| ./a.out 89k | Error - improper formatting | yes |
| ./a.out | Error - improper formatting | yes |