Project 1:

Title:

Left Center Right V.1

Dice Game

Class:

Cis 5 - 41367

Due Date:

February 7, 2021

Author:

Aamir Khan

Introduction:

Title: Left Center Right V.1

Goal: The goal of the game is to be the last one standing with tokens. This game may be played for fun or may be played with consequences on the line that make for a more competitive game.

Rules: As of now, only 3 players are allowed. Each player will take a turn rolling 3 dice to begin with, and on successive turns, will roll as many dice as they have tokens. To Player 1's left is Player 2 and to their right is Player 3. To Player 2's left is Player 3 and to their right is Player 1. To Player 3's left is Player 1 and to their right is Player 2.

1. Roll a 4 - Give a token to your opponent on the left.

2. Roll a 5 - A token is lost to the center.

3. Roll a 6 - Give a token to your opponent on the right.

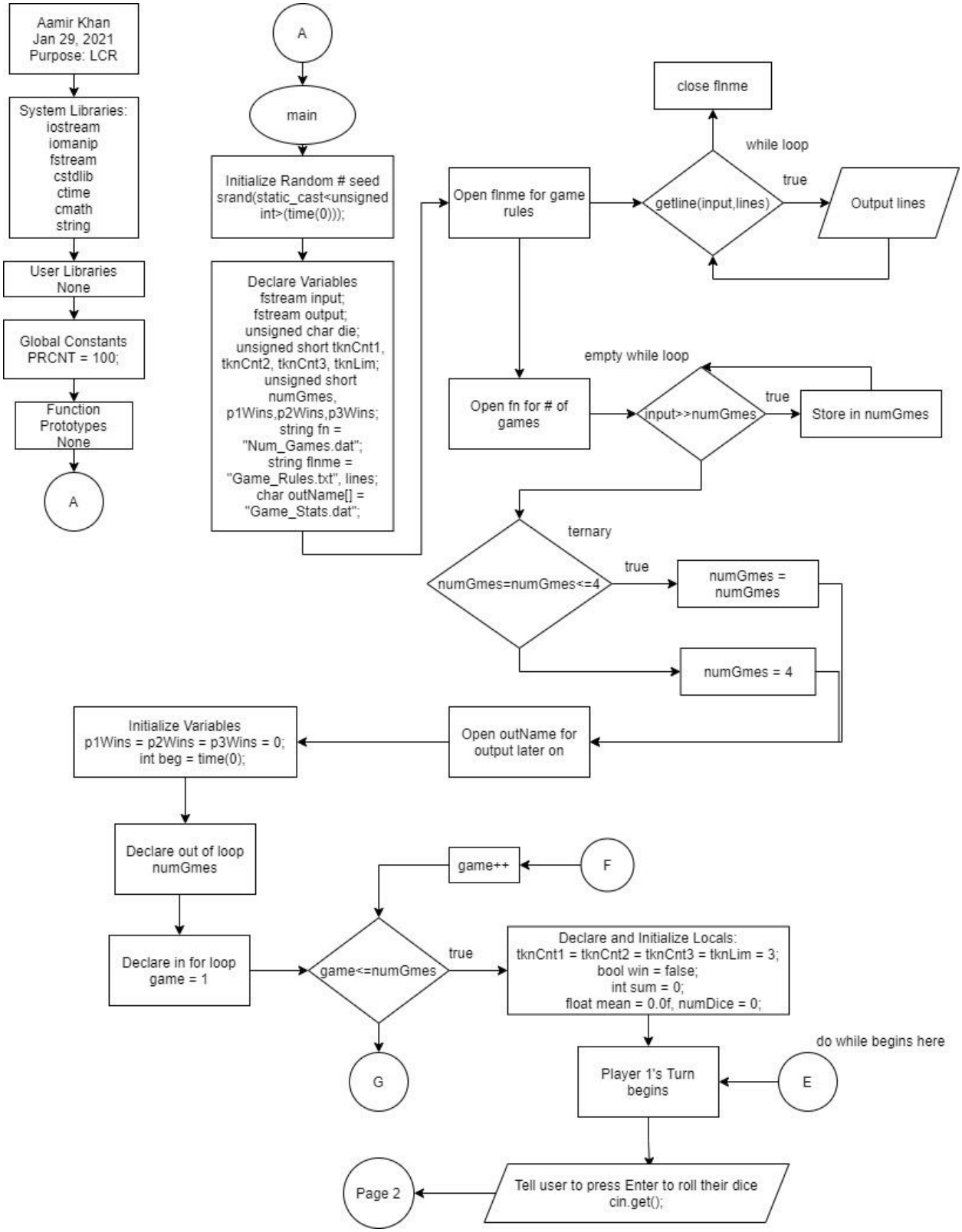
4. Roll a 1-3 - The game goes on and no token is lost for that die.

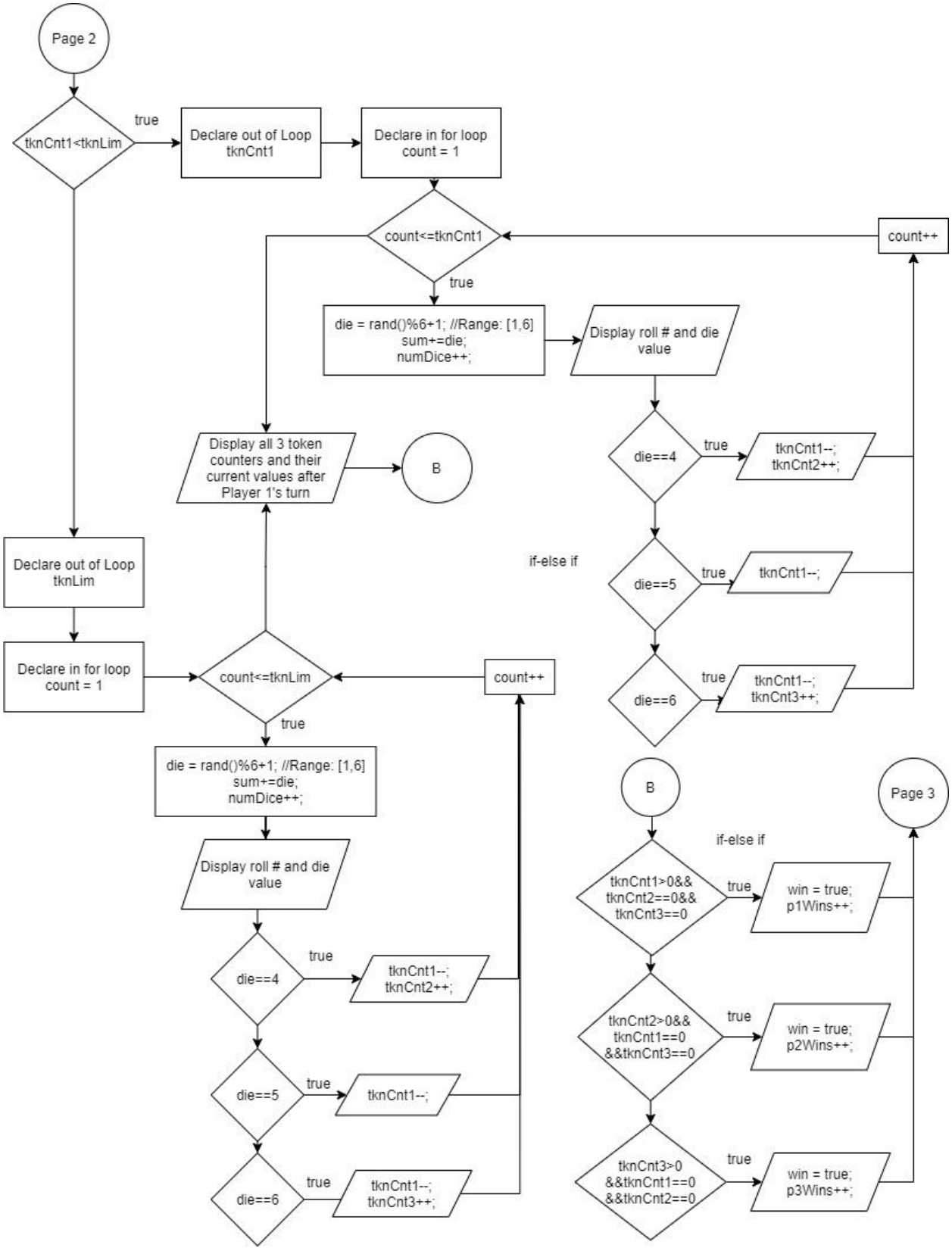
Summary:

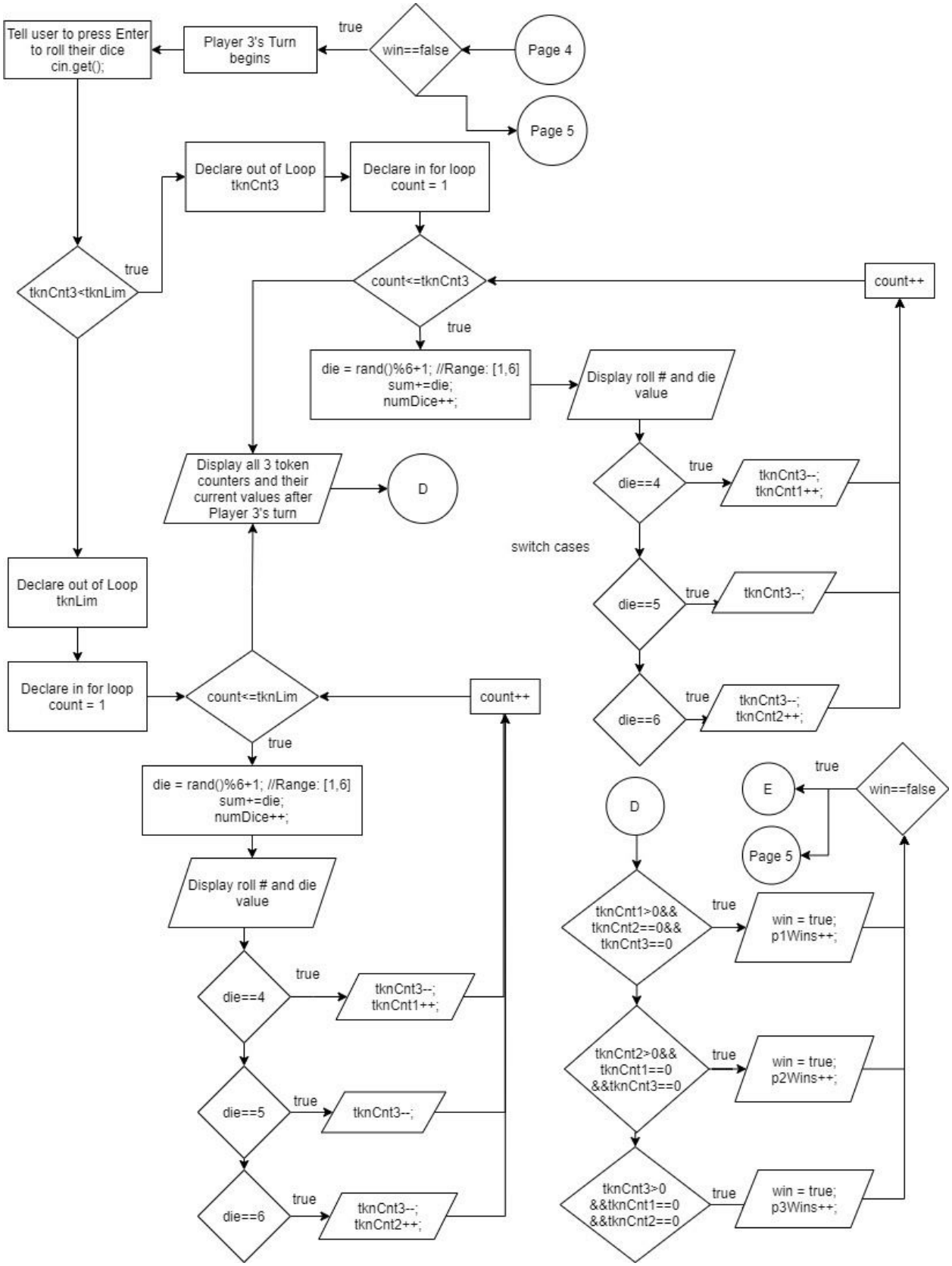
Project Length: About 195 Lines

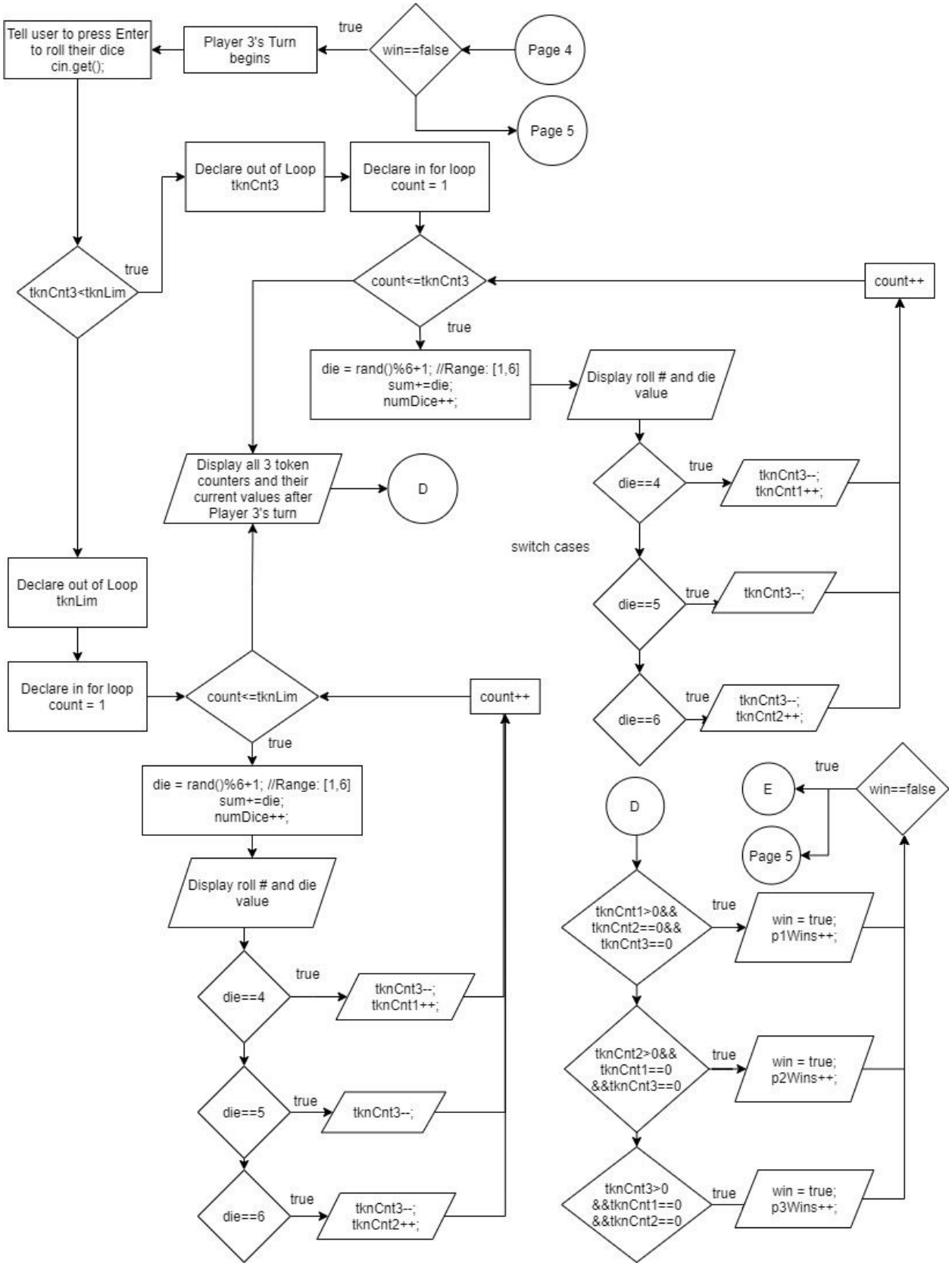
For this project, I used a fairly simple approach. The game takes advantage of for loops and dependent if statements for the calculation side of things. A do-while is used for repetition as needed and a number of games may be played on one run. As far as concepts are concerned, I used various branching constructs and data types that I picked up on throughout Gaddis, chapters 1-5. I could've improved on this version by adding more players and possibly a preferred starting point with respect to tokens or chips.

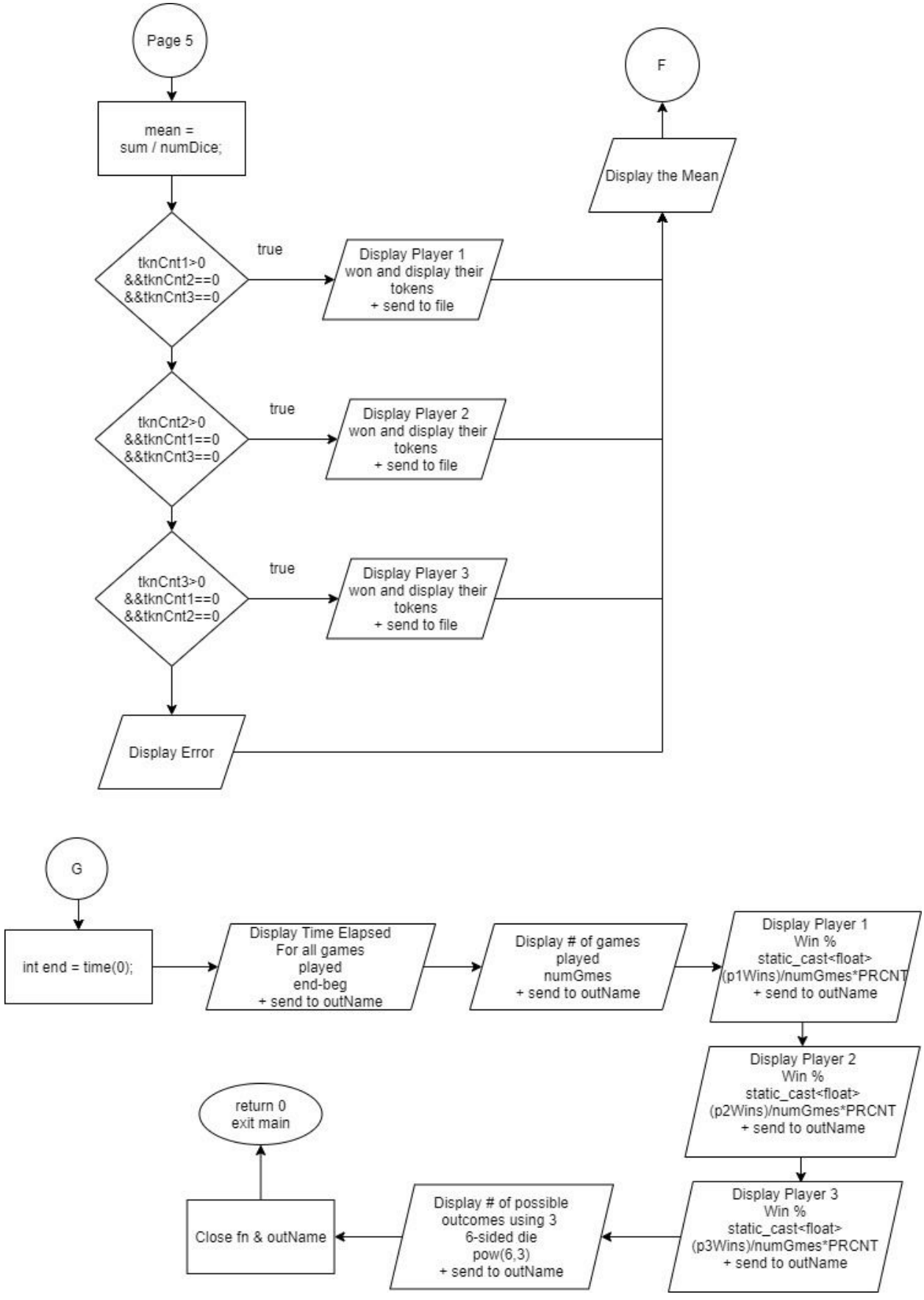
LCR Flowchart:




Page 2
tknCnt1<tknLim
true
Declare out of Loop tknCnt1
Declare in for loop count = 1
count<=tknCnt1
count++
true
die = rand()%6+1; //Range: [1,6]
sum+=die;
numDice++;
Display roll # and die value
Display all 3 token counters and their current values after Player 1's turn
B
die==4
true
tknCnt1--;
tknCnt2++;
if-else if
die==5
true
tknCnt1--;
die==6
true
tknCnt1--;
tknCnt3++;
Declare out of Loop tknLim
Declare in for loop count = 1
count<=tknLim
count++
true
die = rand()%6+1; //Range: [1,6]
sum+=die;
numDice++;
Display roll # and die value
die==4
true
tknCnt1--;
tknCnt2++;
die==5
true
tknCnt1--;
die==6
true
tknCnt1--;
tknCnt3++;
B
Page 3
if-else if
tknCnt1>0&&
tknCnt2==0&&
tknCnt3==0
true
win = true;
p1Wins++;
tknCnt2>0&&
tknCnt1==0
&&tknCnt3==0
true
win = true;
p2Wins++;
tknCnt3>0
&&tknCnt1==0
&&tknCnt2==0
true
win = true;
p3Wins++;


Tell user to press Enter to roll their dice cin.get();
Player 3's Turn begins
true
win==false
Page 4
Page 5
Declare out of Loop tknCnt3
Declare in for loop count = 1
true
tknCnt3<tknLim
count<=tknCnt3
count++
true
die = rand()%6+1; //Range: [1,6]
sum+=die;
numDice++;
Display roll # and die value
Display all 3 token counters and their current values after Player 3's turn
D
die==4
true
tknCnt3--;
tknCnt1++;
switch cases
die==5
true
tknCnt3--;
die==6
true
tknCnt3--;
tknCnt2++;
Declare out of Loop tknLim
Declare in for loop count = 1
count<=tknLim
count++
true
die = rand()%6+1; //Range: [1,6]
sum+=die;
numDice++;
Display roll # and die value
die==4
true
tknCnt3--;
tknCnt1++;
die==5
true
tknCnt3--;
die==6
true
tknCnt3--;
tknCnt2++;
D
true
E
win==false
Page 5
tknCnt1>0&&
tknCnt2==0&&
tknCnt3==0
true
win = true;
p1Wins++;
tknCnt2>0&&
tknCnt1==0
&&tknCnt3==0
true
win = true;
p2Wins++;
tknCnt3>0
&&tknCnt1==0
&&tknCnt2==0
true
win = true;
p3Wins++;


Tell user to press Enter to roll their dice
cin.get();
Player 3's Turn begins
true
win==false
Page 4
Page 5
Declare out of Loop tknCnt3
Declare in for loop count = 1
true
tknCnt3<tknLim
count<=tknCnt3
count++
true
die = rand()%6+1; //Range: [1,6]
sum+=die;
numDice++;
Display roll # and die value
Display all 3 token counters and their current values after Player 3's turn
D
die==4
true
tknCnt3--;
tknCnt1++;
switch cases
die==5
true
tknCnt3--;
die==6
true
tknCnt3--;
tknCnt2++;
Declare out of Loop tknLim
Declare in for loop count = 1
count<=tknLim
count++
true
die = rand()%6+1; //Range: [1,6]
sum+=die;
numDice++;
Display roll # and die value
die==4
true
tknCnt3--;
tknCnt1++;
die==5
true
tknCnt3--;
die==6
true
tknCnt3--;
tknCnt2++;
D
true
E
win==false
Page 5
tknCnt1>0&&
tknCnt2==0&&
tknCnt3==0
true
win = true;
p1Wins++;
tknCnt2>0&&
tknCnt1==0
&&tknCnt3==0
true
win = true;
p2Wins++;
tknCnt3>0
&&tknCnt1==0
&&tknCnt2==0
true
win = true;
p3Wins++;


Page 5
mean =
sum / numDice;
tknCnt1>0
&&tknCnt2==0
&&tknCnt3==0
true
Display Player 1
won and display their
tokens
+ send to file
tknCnt2>0
&&tknCnt1==0
&&tknCnt3==0
true
Display Player 2
won and display their
tokens
+ send to file
tknCnt3>0
&&tknCnt1==0
&&tknCnt2==0
true
Display Player 3
won and display their
tokens
+ send to file
Display Error
F
Display the Mean
G
int end = time(0);
Display Time Elapsed
For all games
played
end-beg
+ send to outName
Display # of games
played
numGmes
+ send to outName
Display Player 1
Win %
static_cast<float>
(p1Wins)/numGmes*PRCNT
+ send to outName
Display Player 2
Win %
static_cast<float>
(p2Wins)/numGmes*PRCNT
+ send to outName
Display Player 3
Win %
static_cast<float>
(p3Wins)/numGmes*PRCNT
+ send to outName
Display # of possible
outcomes using 3
6-sided die
pow(6,3)
+ send to outName
Close fn & outName
return 0
exit main

Pseudo-Code:

```
/*
* File:   main.cpp
 * Author: Aamir
 * Created on January 26, 2021, 6:20 PM
 * Purpose:  Adding onto V.6- LCR V.7
 * Using a bool to continue repetitions as needed + testing for a win + switch cases
 */

//System Libraries
//I/O Library
//Formatting Library
//String Library
//File Library
//Rand # Generator
//Time set to seed
//Math Library
//Namespace std of system libraries

//User Libraries

//Global Constants
//Percentage CNV

//Main -> Execution Begins Here
        //Initialize the Random # Generator Seed
        //Declare Variables
        //Input.open/close() functions
        //Output.open/close() functions
        //Die value
        //Token counters for players 1-3 and token limit
        //store # of games read from file and wins counter
        //# of games file
        //fn file and variable that will store the text
        //outName file for game stats
        //Open game rules file, read in contents + display, close file
        //Open fn for input & outName for output

        //Input the # of games and loop through file to read in content
        //Limit the # of games to no more than 4

        //Initialize all win counters to 0
        //Declare and Initialize beg variable to keep track of time -> beg = time(0);
```

```
//Play the game specified number of times through looping
        //Initialize Local Variables
        //Token Counters, bool win, sum, mean, numDice

        //Loop through Player 1-3's turns while no one has won
                //Loop for rolling dice
                        //If Player has less than 3 tokens, roll for as many tokens you have
                        //Else roll 3 dice for 3 or more tokens
                        //Based on what die is rolled and what Player rolled the die, token
                        //counters will be incremented/decremented
                        //All the dice will also be added to the sum and numDice
                        //incremented for each die rolled to calculate the mean at the end

                        //All token counter values will be display after dice are rolled
                        //Test for a win

                        //For Player 2, if win==true, the game ends before their turn
                        //If win==false, their turns continue and rolling dice continues
                        //For Player 3, if win==true, the game ends before their turn
                        //If win==false, their turns continue and rolling dice continues
                        //This rolling and testing for a win process repeats
                        //for each player until someone has won.
                        //Then the loop for player turns is exited

        //Calculate the mean die value rolled
        //Display the winner and their leftover tokens + output to file
        //Display the mean die value rolled + output to file
        //This will continue until all games are played and the for loop is exited

//Declare and Initialize end variable to keep track of the time elapsed since beg -> end = time(0);
//Display some game info such as the time elapsed for all games -> (end-beg);
//Send these stats and game info to a file-> (Game_Stats.dat)
//Close the remaining two files and exit main!
```

Stages of Development:

Version 1 - Rolling1Die

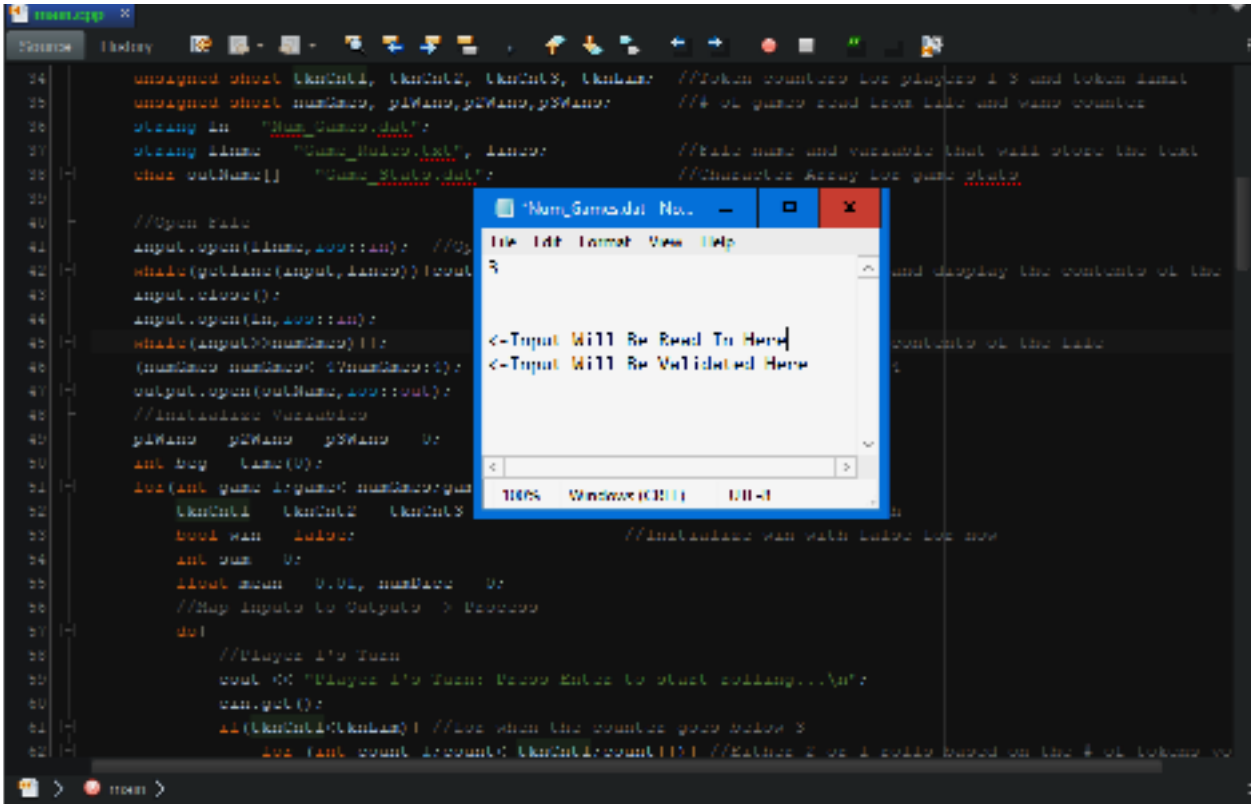
This was where my game began. The idea behind this version was to get a single working die and some stats to come with. Not much was covered here except using a random number generator to get a random number between 1 and 6.

Version 2 - Rolling3Die

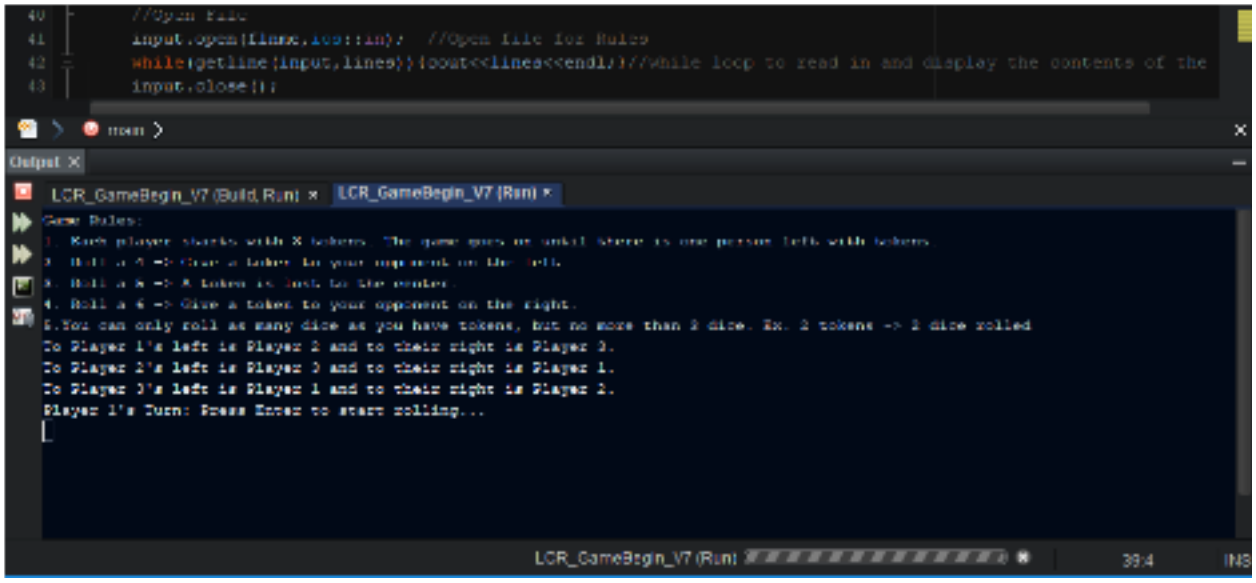
This was where I got 3 working dice to go with my game. I added onto the previous version to get three random values, each between 1 and 6 to simulate rolling 3 dice. (Since die is declared as a char, it's necessary to type cast each time we roll a die as an int.)

Version 3 - TokenCounters
In this version, I took what I wrote in previous versions and translated it over to a token counter for one player. After rolling 3 dice a number of times through looping, I returned back the percentage out of the number of loops that a 4,5, and 6 appeared. I also took into account when the counter fell below 3 and what would happen if that were the case. The basic idea was implemented in this version that would then be implemented in my final version as well.
Version 4 - TokenCounters
This version prioritized creating 3 working counters, and actually getting started on the token calculation side of things. In this version, one round is experienced, but the 3 token counters are incremented and decremented appropriately based on what die was rolled and who rolled it. With each turn, current token counts would then be displayed. If Player 1 rolled a 4, Player 1 would lose a token and it would go to Player 2. When a 5 is rolled, the token is lost for good, and when a 6 is rolled, the token is lost and given to the opponent to the right. All of this was implemented in this version, and would then go on to version 5 alongside looping. This version utilized dependent if statements to accomplish logic.
Version 5 - GameBegin
This version introduced looping into the equation by using a do-while loop. The condition wasn't final, but it was to test whether looping worked with everything present. The focus for this version was to introduce file input and reading. Additionally, a while loop was used to read in the contents of the file. This input was then validated using a ternary operator. In this version, I read in the number of players playing, but this would later change to the number of games that wanted to be played by the final version. I also read in the contents of another file using a while loop that then displayed each line from the .txt file (Game_Rules.txt). Not much was done in this version besides looping and adding file input.
Version 6 - GameBegin
In this version, a boolean was used to continue repetitions as needed within the do-while loop. Up until now, every player's token calculations were done using dependent ifs, so switch cases were used as well in place of some dependent ifs. Independent if statements are also used after Player 1's turn as well as Player 2's turn to determine whether the game should be continued or not. At the very end, the winner is displayed using a nested ternary operator. Versions 1-6 now cover most of what we've learned in CIS-5 Chs. 1-5, aside from some missing data types and other pieces that will added onto version 7 as finishing touches. From here on, the game's logic is complete, but other necessary components needed for the project will be thrown in.
Version 7 - GameBegin
The first thing I changed in this version was reading in the number of games as opposed to the number of players from previous versions. I also introduce file output via some game information that's sent to a file (Game_Stats.dat) which can be accessed following the game's completion. This means that the game can now be played multiple times, but no more than 4 in one run. For the sake of more data, I also threw in a mean calculator that calculates the mean die value rolled throughout one complete game. The information that's to be sent to a file includes: the time elapsed for all games played in seconds, the number of games played, the win percentages of all players, and the number of possible outcomes one could have in using 3 6-sided dice. This version concludes the game's stages of development and marks the end of LCR V.1.

Sample Screenshots:



The image above demonstrates input (# of games) being read in and validated in our program.



The image above shows the rules being read in from a file and displayed in the console at start.

```
Time Elapsed for all games played: 276 seconds
Number of Games Played: 3
Player 1 Percentage of Wins: 66.67%
Player 2 Percentage of Wins: 0.00%
Player 3 Percentage of Wins: 33.33%
Number of Possible Combinations using 3 six-sided die: 216.00
RUN SUCCESSFUL (total time: 4m 36s)
```

This is an example of what one run along with stats may look like. (Results will differ.)

```
Player 1 Current Token Count = 0
Player 2 Current Token Count = 0
Player 3 Current Token Count = 2
Player 3 won with 2 tokens!
Mean = 3.33
```

This is an example of what a win would look like. All other token counters are zero. The mean value rolled that game is also given at the end of each game or win.

```
//Displaying Winner and some stats
(tknCnt1>0&&tknCnt2==0&&tknCnt3==0?cout<<"Player 1 won with "<<tknCnt1<< " tokens!\n":
 tknCnt2>0&&tknCnt1==0&&tknCnt3==0?cout<<"Player 2 won with "<<tknCnt2<< " tokens!\n":
 tknCnt3>0&&tknCnt1==0&&tknCnt2==0?cout<<"Player 3 won with "<<tknCnt3<<" tokens!\n":
 cout<<"Error\n");
```

An instance of a nested ternary operator that's used to display the winner and their ending token count. This conditional is mentioned after each turn to check for a win.

```
//Sending some stats to a file
output<< "Time Elapsed for all games played: "<<end-beg<<" seconds"<< endl;
output<<"Number of Games Played: "<<numGmes<<endl;
output<<"Player 1 Percentage of Wins: "<<static_cast<float>(p1Wins)/numGmes*PRCNT<<"%"<<endl;
output<<"Player 2 Percentage of Wins: "<<static_cast<float>(p2Wins)/numGmes*PRCNT<<"%"<<endl;
output<<"Player 3 Percentage of Wins: "<<static_cast<float>(p3Wins)/numGmes*PRCNT<<"%"<<endl;
output<<"Number of Possible Combinations using 3 six-sided die: "<<pow(6,3);
```

The image above shows data being outputted to a file (Game_Stats.dat). (Viewable Afterwards.)

Project 1 Check Off Sheet:

Ch.	Section	Topic	Where Line #"s	Pts	Notes
2	2	cout			
	3	libraries	Begins @ Line 10	8	iostream, iomanip, cmath, cstdlib, fstream, string, ctime
	4	variables/literals			No variables in global area, failed project!
	5	Identifiers			
	6	Integers	Line 30	3	Setting seed using unsigned int
	7	Characters	Line 34	3	
	8	Strings	Line 39	3	Character string for file name
	9	Floats No Doubles	Line 166	3	Using doubles will fail the project, floats OK!
	10	Bools	Line 54/161	4	Used as do-while condition
	11	Sizeof *****			
	12	Variables 7 characters or less			All variables <= 7 characters
	13	Scope ***** No Global Variables			
	14	Arithmetic operators			
	15	Comments 20%+	Throughout the game	5	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	Line 67	4	casting char die as int
	6	Multiple assignment *****			

	7	Formatting output	Line 179	4	for displaying end of game stats
	8	Strings	Line 36	3	Storing file names
	9	Math Library	Line 185	4	Calling pow function
	10	Hand tracing ******			
4	1	Relational Operators			
	2	if	Line 89	4	Independent if to test for a win yet
	4	If-else	Line 62 & 72	4	To roll either 3 or less based on token counter
	5	Nesting	Lines 62-70	4	Token counter operations
	6	If-else-if	Lines 68-70	4	Calculations for token counters
	7	Flags *****			
	8	Logical operators	Line 86-88	4	Checking for a win to finish game
	11	Validating user input	Line 47	4	Reading # of games from file and check if <=4
	13	Conditional Operator	Line 164	4	Nested ternary for announcing winner
	14	Switch	Line 99	4	Token counter operations
5	1	Increment/Decrement	Line 68	4	Incrementing/Decrementing counters
	2	While	Line 46	4	Read in file content
	5	Do-while	Line 58	4	Repeat turns as long as no one won
	6	For loop	Line 52	4	Keep track of # of games
	11	Files input/output both	Lines 42-48 & 187	8	Read file inputs and send game info at the end to file
	12	No breaks in loops ******			Failed Project if included
****** Not required to show			Total	100	

Program Code:

(Note: The program is given here for the sake of the write-up's style. Refer to the actual program folder for a better look.)

```
//System Libraries
#include <iostream>  //I/O Library
#include <iomanip>  //Formatting Library
#include <string>      //String Library
#include <fstream>   //File Library
#include <cstdlib>    //Rand # Generator
#include <ctime>      //Time set to seed
#include <cmath>     //Math Library
using namespace std;

//User Libraries
//Global Constants
//Math, Science, Universal, Conversions, High Dimensioned Arrays
const short PRCNT = 100;
//Function Prototypes
//Execution Begins Here
int main(int argc, char** argv) {
   //Initialize the Random Number Seed
   srand(static_cast<unsigned int>(time(0)));        //Set Random # Generator
   //Declare Variables
   fstream input;                              //Input.open/close() functions
   fstream output;                             //Output.open/close() functions
   unsigned char die;                          //Die values rolled by each player
   unsigned short tknCnt1, tknCnt2, tknCnt3, tknLim;  //Token counters for players 1-3 and limit
   unsigned short numGmes, p1Wins,p2Wins,p3Wins;  //stores # of games, 3 win counters
   string fn = "Num_Games.dat";
   string flnme = "Game_Rules.txt", lines;          //File name and variable that will store the text
   char outName[] = "Game_Stats.dat";               //Character Array for game stats

   //Open File
   input.open(flnme,ios::in);  //Open file for Rules
   while(getline(input,lines)){cout<<lines<<endl;}
   //while loop to read in and display the contents of the file
   input.close();
   input.open(fn,ios::in);
   while(input>>numGmes){};           //Empty while loop to read in the contents of the file
   (numGmes=numGmes<=4?numGmes:4);     //Only allowing games to be reach 4
   output.open(outName,ios::out);                  //For when we do stats at the end
```

```
//Initialize Variables
p1Wins = p2Wins = p3Wins = 0;
int beg = time(0);
for(int game=1;game<=numGmes;game++){
   tknCnt1 = tknCnt2 = tknCnt3 = tknLim = 3; //Start with 3 tokens each
   bool win = false;                         //Initialize win with false for now
   int sum = 0;
   float mean = 0.0f, numDice = 0;
   //Map Inputs to Outputs -> Process
   do{
      //Player 1's Turn
      cout << "Player 1's Turn: Press Enter to start rolling...\n";
      cin.get();
      if(tknCnt1<tknLim){ //for when the counter goes below 3
         for (int count=1;count<=tknCnt1;count++){
         //Either 2 or 1 rolls based on the # of tokens you have
            die = rand()%6+1; //Range: [1,6]
            sum+=die;
            numDice++;
            cout << "Roll "<<count<< " = " <<static_cast<int>(die)<<endl;
            if (die==4){tknCnt1--;tknCnt2++;}      //Token is given to the left or Player 2
            else if (die==5) tknCnt1--;           //Token is placed in the center and is lost for good
            else if (die==6){tknCnt1--;tknCnt3++;} //Token is given to the right or Player 3
         }
      }else{ //for when the counter is 3 or more
         for (int count=1;count<=tknLim;count++){ //3 rolls for having 3 or more tokens
            die = rand()%6+1; //Range: [1,6]
            sum+=die;
            numDice++;
            cout << "Roll "<<count<< " = " << static_cast<int>(die)<<endl;
            if (die==4){tknCnt1--;tknCnt2++;}      //Token is given to the left or Player 2
            else if (die==5) tknCnt1--;           //Token is placed in the center and is lost for good
            else if (die==6){tknCnt1--;tknCnt3++;} //Token is given to the right or Player 3
         }
      }
      cout << "Player 1 Current Token Count = " << tknCnt1<<endl;
      cout << "Player 2 Current Token Count = " << tknCnt2<<endl;
      cout << "Player 3 Current Token Count = " << tknCnt3<<endl;
      if(tknCnt1>0&&tknCnt2==0&&tknCnt3==0){win = true;p1Wins++;}
      else if(tknCnt2>0&&tknCnt1==0&&tknCnt3==0){win = true;p2Wins++;}
```

```
    else if(tknCnt3>0&&tknCnt1==0&&tknCnt2==0){win = true;p3Wins++;}
    if(win==false){ //Independent if to see if win==false to continue turns
    //Player 2's Turn
    cout << "Player 2's Turn: Press Enter to start rolling...\n";
    cin.get();
    if(tknCnt2<tknLim){ //for when the counter goes below 3
       for (int count=1;count<=tknCnt2;count++){
       //Either 2 or 1 rolls based on the # of tokens you have
          die = rand()%6+1; //Range: [1,6]
          sum+=die;
          numDice++;
          cout << "Roll "<<count<< " = " <<static_cast<int>(die)<<endl;
          switch(die){
             case 4:tknCnt2--;tknCnt3++;break;//Token is given to the left or Player 3
             case 5:tknCnt2--;break;          //Token is placed in the center and is lost for good
             case 6:tknCnt2--;tknCnt1++;break;//Token is given to the right or Player 1
          }
       }
    }else{ //for when the counter is 3 or more
       for (int count=1;count<=tknLim;count++){ //3 rolls for having 3 or more tokens
          die = rand()%6+1; //Range: [1,6]
          sum+=die;
          numDice++;
          cout << "Roll "<<count<< " = " <<static_cast<int>(die)<<endl;
          switch(die){
             case 4:tknCnt2--;tknCnt3++;break;//Token is given to the left or Player 3
             case 5:tknCnt2--;break;          //Token is placed in the center and is lost for good
             case 6:tknCnt2--;tknCnt1++;break;//Token is given to the right or Player 1
          }
       }
    }
    cout << "Player 1 Current Token Count = " << tknCnt1<<endl;
    cout << "Player 2 Current Token Count = " << tknCnt2<<endl;
    cout << "Player 3 Current Token Count = " << tknCnt3<<endl;
    if(tknCnt1>0&&tknCnt2==0&&tknCnt3==0){win = true;p1Wins++;}
    else if(tknCnt2>0&&tknCnt1==0&&tknCnt3==0){win = true;p2Wins++;}
    else if(tknCnt3>0&&tknCnt1==0&&tknCnt2==0){win = true;p3Wins++;}
}
if (win==false){
    //Player 3's Turn
```

```
        cout << "Player 3's Turn: Press Enter to start rolling...\n";
        cin.get();
        if(tknCnt3<tknLim){ //for when the counter goes below 3
           for (int count=1;count<=tknCnt3;count++){
           //Either 2 or 1 rolls based on the # of tokens you have
              die = rand()%6+1; //Range: [1,6]
              sum+=die;
              numDice++;
              cout << "Roll "<<count<< " = " <<static_cast<int>(die)<<endl;
              switch(die){
                 case 4:tknCnt3--;tknCnt1++;break;//Token is given to the left or Player 1
                 case 5:tknCnt3--;break;          //Token is placed in the center and is lost for good
                 case 6:tknCnt3--;tknCnt2++;break;//Token is given to the right or Player 2
              }
           }
        }else{ //for when the counter is 3 or more
           for (int count=1;count<=tknLim;count++){ //3 rolls for having 3 or more tokens
              die = rand()%6+1; //Range: [1,6]
              sum+=die;
              numDice++;
              cout << "Roll "<<count<< " = " <<static_cast<int>(die)<<endl;
              switch(die){
                 case 4:tknCnt3--;tknCnt1++;break;//Token is given to the left or Player 1
                 case 5:tknCnt3--;break;          //Token is placed in the center and is lost for good
                 case 6:tknCnt3--;tknCnt2++;break;//Token is given to the right or Player 2
              }
           }
        }
        cout << "Player 1 Current Token Count = " << tknCnt1<<endl;
        cout << "Player 2 Current Token Count = " << tknCnt2<<endl;
        cout << "Player 3 Current Token Count = " << tknCnt3<<endl;
        if(tknCnt1>0&&tknCnt2==0&&tknCnt3==0){win = true;p1Wins++;}
        else if(tknCnt2>0&&tknCnt1==0&&tknCnt3==0){win = true;p2Wins++;}
        else if(tknCnt3>0&&tknCnt1==0&&tknCnt2==0){win = true;p3Wins++;}
     }
     }while(win==false);
     mean = sum / numDice;
     //Displaying Winner and some stats
     (tknCnt1>0&&tknCnt2==0&&tknCnt3==0?cout<<"Player 1 won with "<<tknCnt1<< "
tokens!\n":
```

```
        tknCnt2>0&&tknCnt1==0&&tknCnt3==0?cout<<"Player 2 won with "<<tknCnt2<< "
tokens!\n":
        tknCnt3>0&&tknCnt1==0&&tknCnt2==0?cout<<"Player 3 won with "<<tknCnt3<<"
tokens!\n":
       cout<<"Error\n");
       cout<<setprecision(2)<<fixed;
       cout<<"Mean = "<<mean<<endl;
       //Send winners to the file
       (tknCnt1>0&&tknCnt2==0&&tknCnt3==0?output<<"Player 1 won with "<<tknCnt1<< "
tokens!\n":
        tknCnt2>0&&tknCnt1==0&&tknCnt3==0?output<<"Player 2 won with "<<tknCnt2<< "
tokens!\n":
        tknCnt3>0&&tknCnt1==0&&tknCnt2==0?output<<"Player 3 won with "<<tknCnt3<<"
tokens!\n":
       output<<"Error\n");
       output<<"Mean = "<<mean<<endl;
   }
   int end = time(0);

   //Displaying some stats
   cout<<fixed<<setprecision(2)<<showpoint;
   cout<< "Time Elapsed for all games played: "<<end-beg<< " seconds"<<endl;
   cout<<"Number of Games Played: "<<numGmes<<endl;
   cout<<"Player 1 Percentage of Wins:
"<<static_cast<float>(p1Wins)/numGmes*PRCNT<<"%"<<endl;
   cout<<"Player 2 Percentage of Wins:
"<<static_cast<float>(p2Wins)/numGmes*PRCNT<<"%"<<endl;
   cout<<"Player 3 Percentage of Wins:
"<<static_cast<float>(p3Wins)/numGmes*PRCNT<<"%"<<endl;
   cout<<"Number of Possible Combinations using 3 six-sided die: "<<pow(6,3);
   //Sending some stats to a file
   output<< "Time Elapsed for all games played: "<<end-beg<<" seconds"<< endl;
   output<<"Number of Games Played: "<<numGmes<<endl;
   output<<"Player 1 Percentage of Wins:
"<<static_cast<float>(p1Wins)/numGmes*PRCNT<<"%"<<endl;
   output<<"Player 2 Percentage of Wins:
"<<static_cast<float>(p2Wins)/numGmes*PRCNT<<"%"<<endl;
   output<<"Player 3 Percentage of Wins:
"<<static_cast<float>(p3Wins)/numGmes*PRCNT<<"%"<<endl;
   output<<"Number of Possible Combinations using 3 six-sided die: "<<pow(6,3);
```

```
    //Exit the Program - Cleanup
    input.close();
    output.close();
    return 0;
}
```