**Project II:**

**Left Center Right**

**CIS-5 Section 41606**

**Name: Jazmyne Patlan**

**Date: February 12th, 2023**

**Introduction**

Left Center Right is a dice game played with tokens. Each player sits in a circle facing the Center pot, which is empty at the start. Each player starts with three tokens and going clockwise they take turns rolling the dice to determine which action to take. This game requires three special dice, with each die having six faces: one with the word “Left,” one with the word “Right,” one with a star, and three with a dot. If the die lands on “Left,” the player has to pass one of their tokens to the player on their left. If the die lands on “Right,” the player has to pass one of their tokens to the player on their right. If the die lands on a star, the player must put a token in the Center pot, where it will remain for the duration of the game. Once in the Center, tokens are no longer in play. If the die lands on a dot, the player is allowed to keep that token.

Though there are three dice, the amount of dice payer’s roll are dependent on the amount of tokens they have. If the player has three or more tokens, they roll all three dice and carry out the action on each one. If the player has two tokens, they only roll two dice and carry out the action on each one. If the player has one token, they only roll one die and carry out the action that it reads.

If the player has no more tokens, they are still in the game, but must skip their turn unless they are passed more tokens from the other players. Players continue to roll and pass tokens until only one player has tokens. Once a player is the only one left with any tokens, they win.

**Summary**

Length: Over 700 lines

Number of Variables: 28

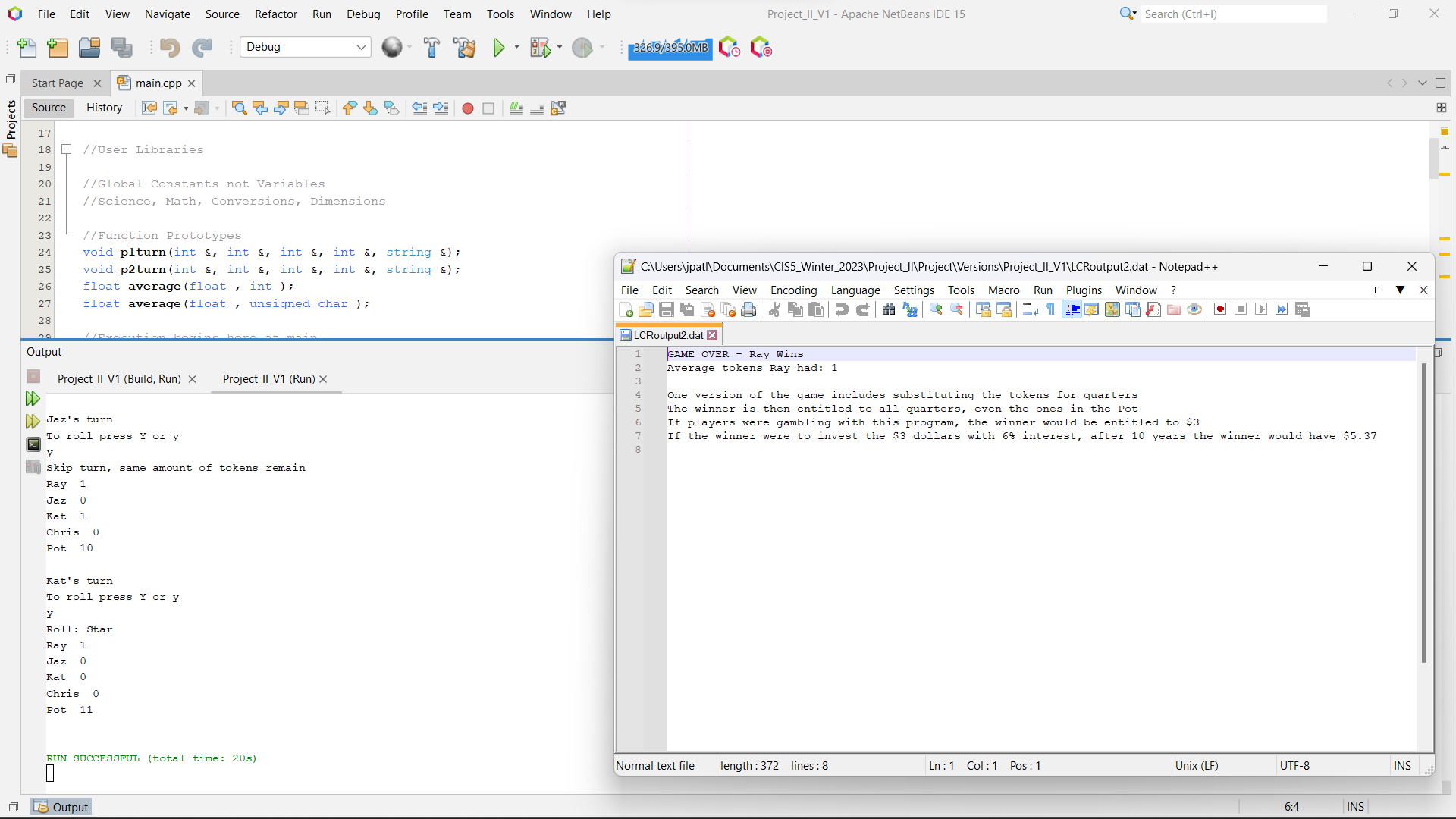
I utilized many concepts needed in Project II. I didn’t utilize all as I’m still not entirely comfortable using some concepts, such as vectors. I managed to incorporate some of the concepts like functions and arrays that I found to be useful and that took my program even further, so I am proud of that.

It took roughly three days to get everything running properly, and though there can be improvements, I’m pleased with what I have come up with so far.

**Description**

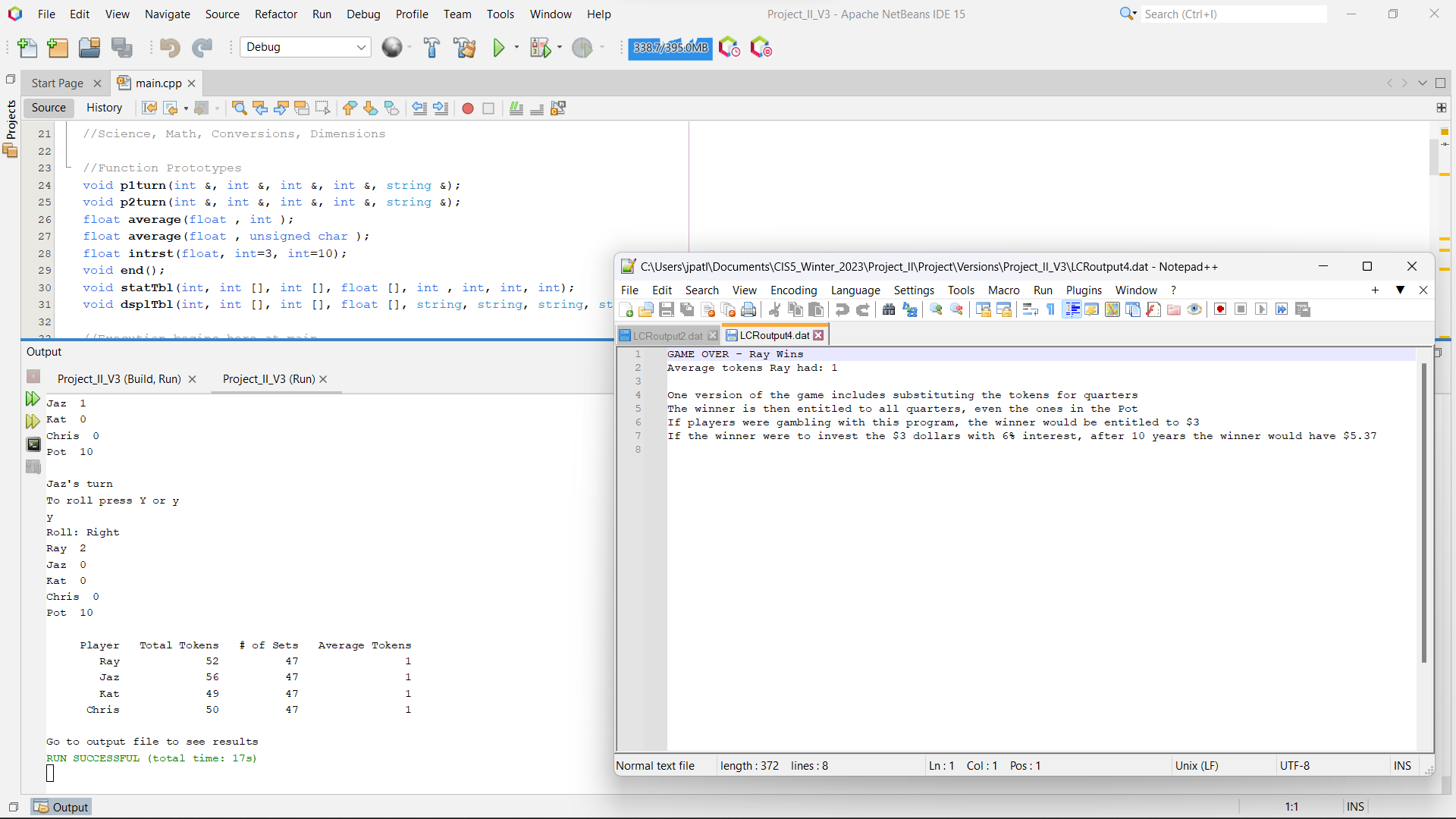
As long as user inputs Y or y to roll, the program will continue until only one player has tokens. The program will then display game statistics and a message directing the user to the output file.

**Version 1 Result:**



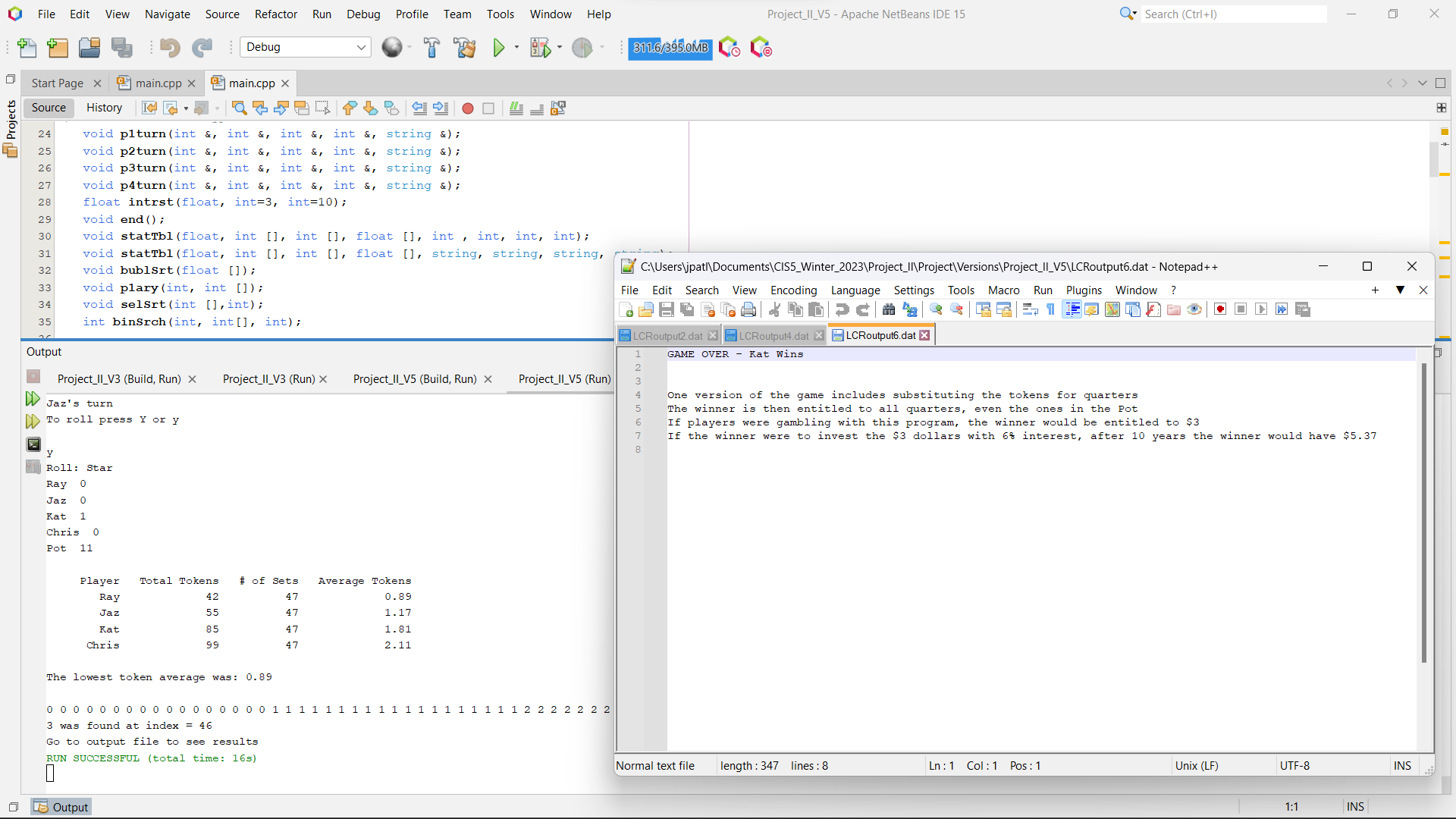
-Adding functions, but outputs the same as final version of Project I

**Version 3 Result:**



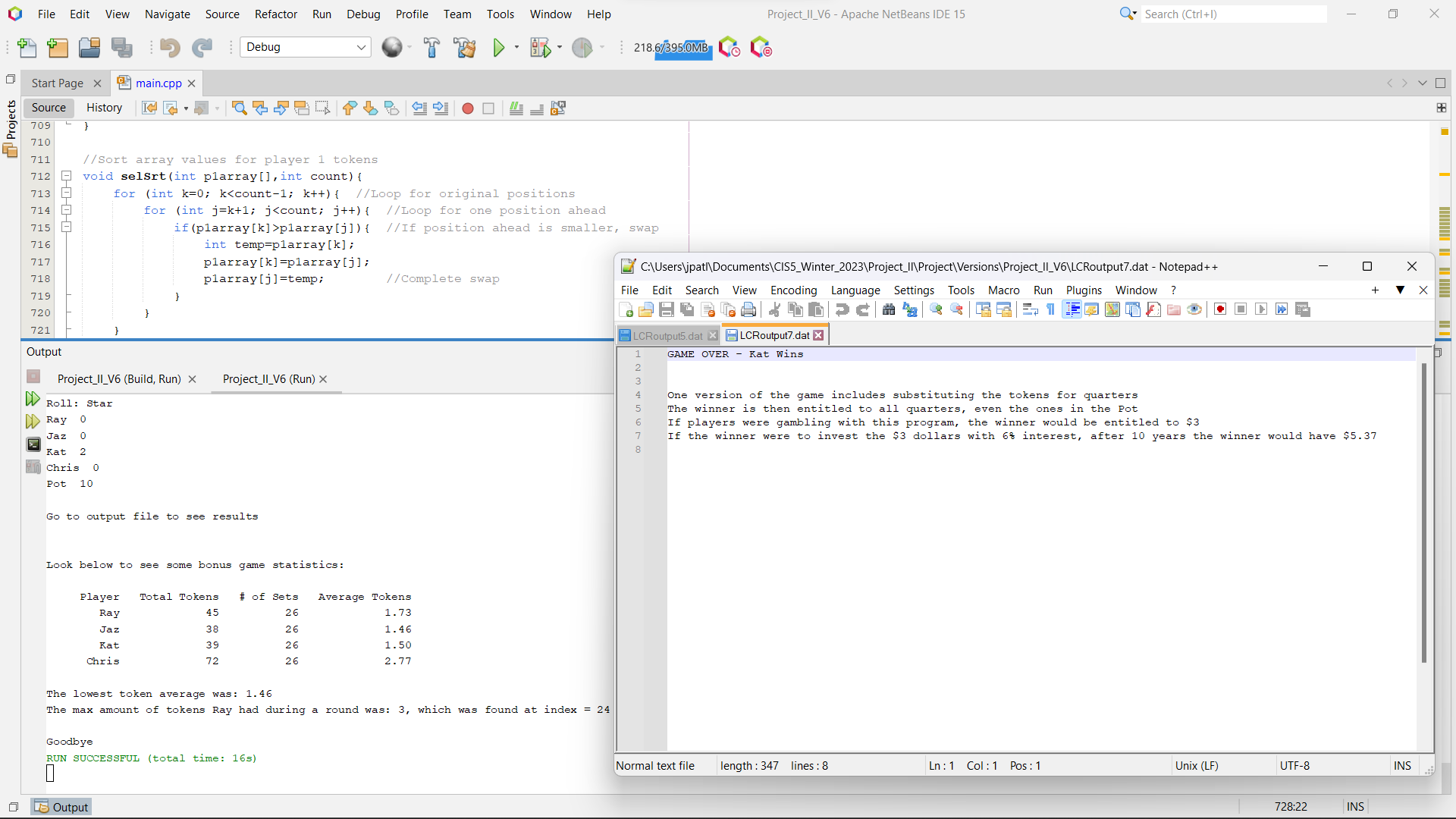
-More functions, adding arrays, and adding a table with player statistics

**Version 5 Result:**

****

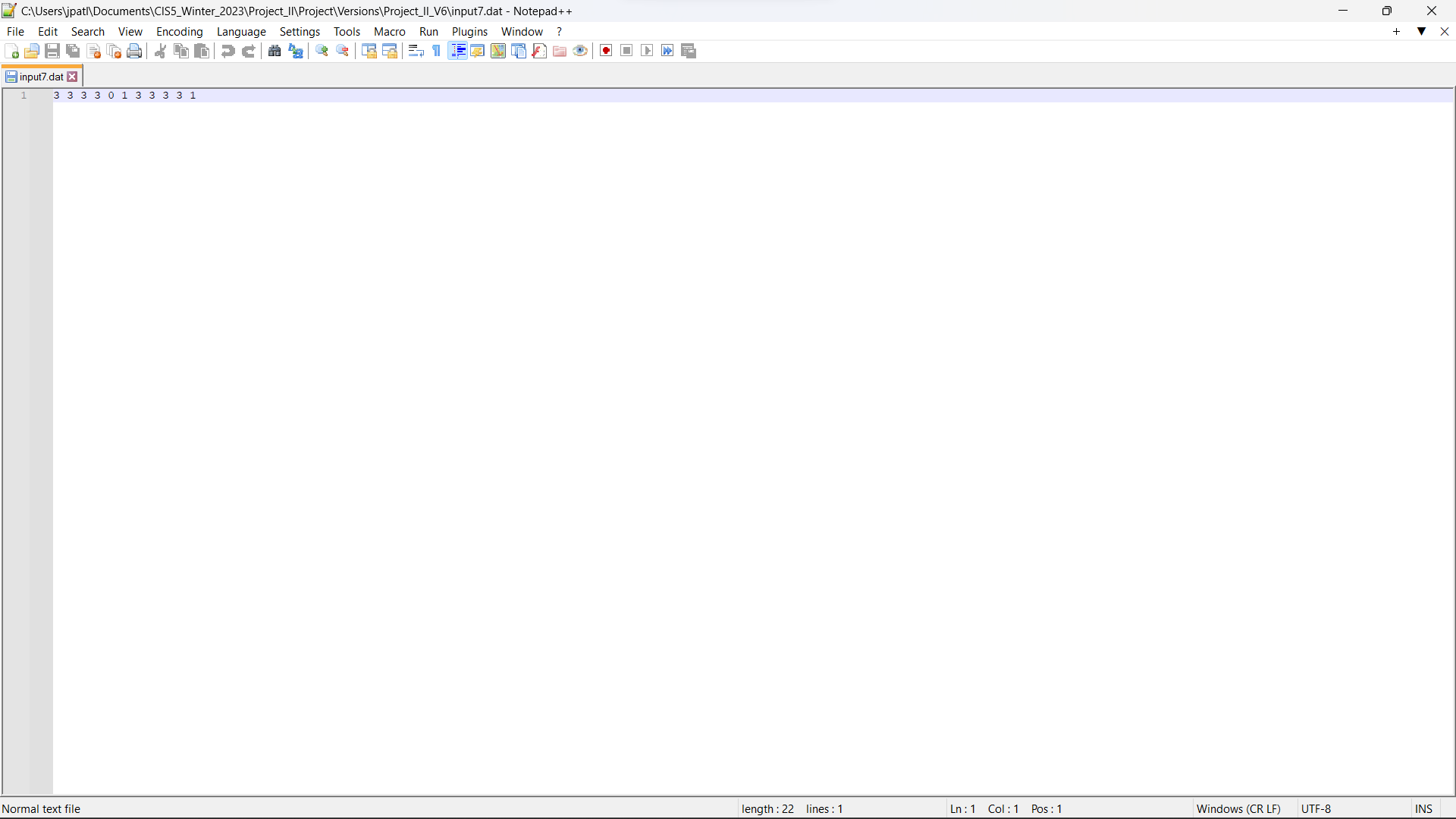
-More functions, including sorts and a search. Displays array of sorted Player 1 tokens from each round to test function and array in order to see if accurate.

**Version 6 Result (Final Project II Version):**

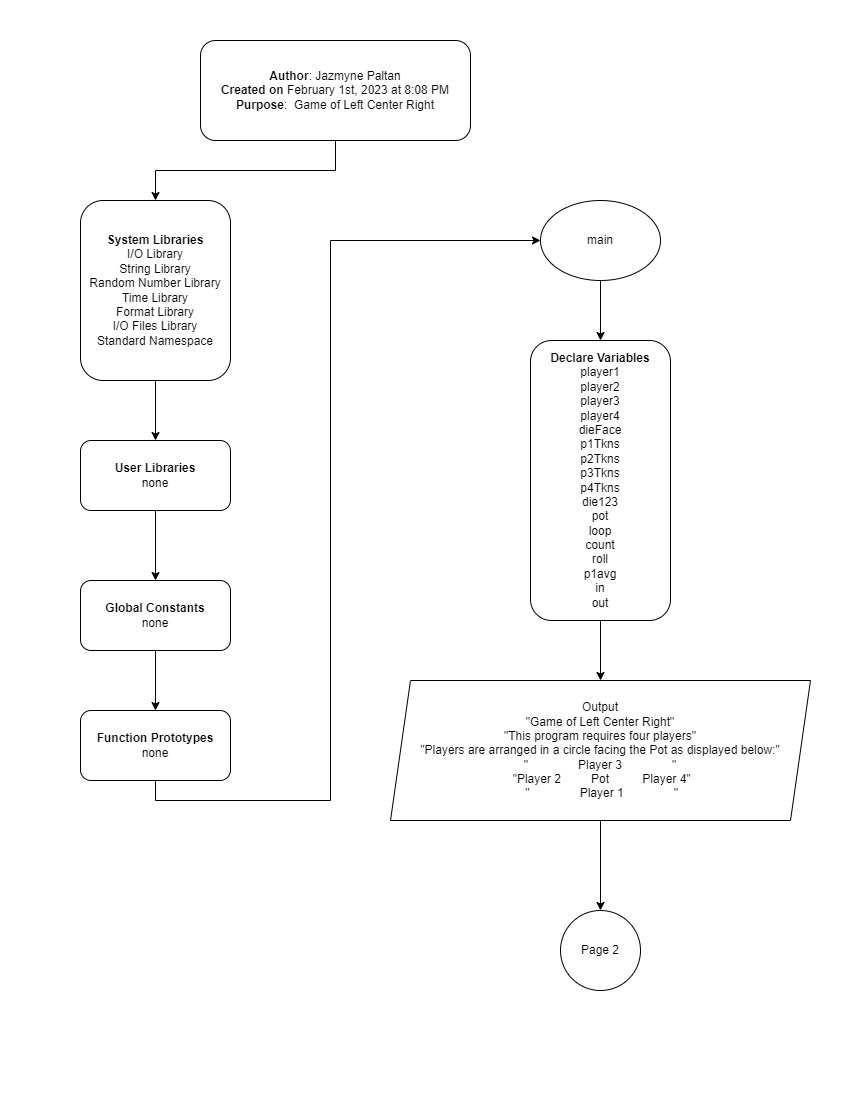


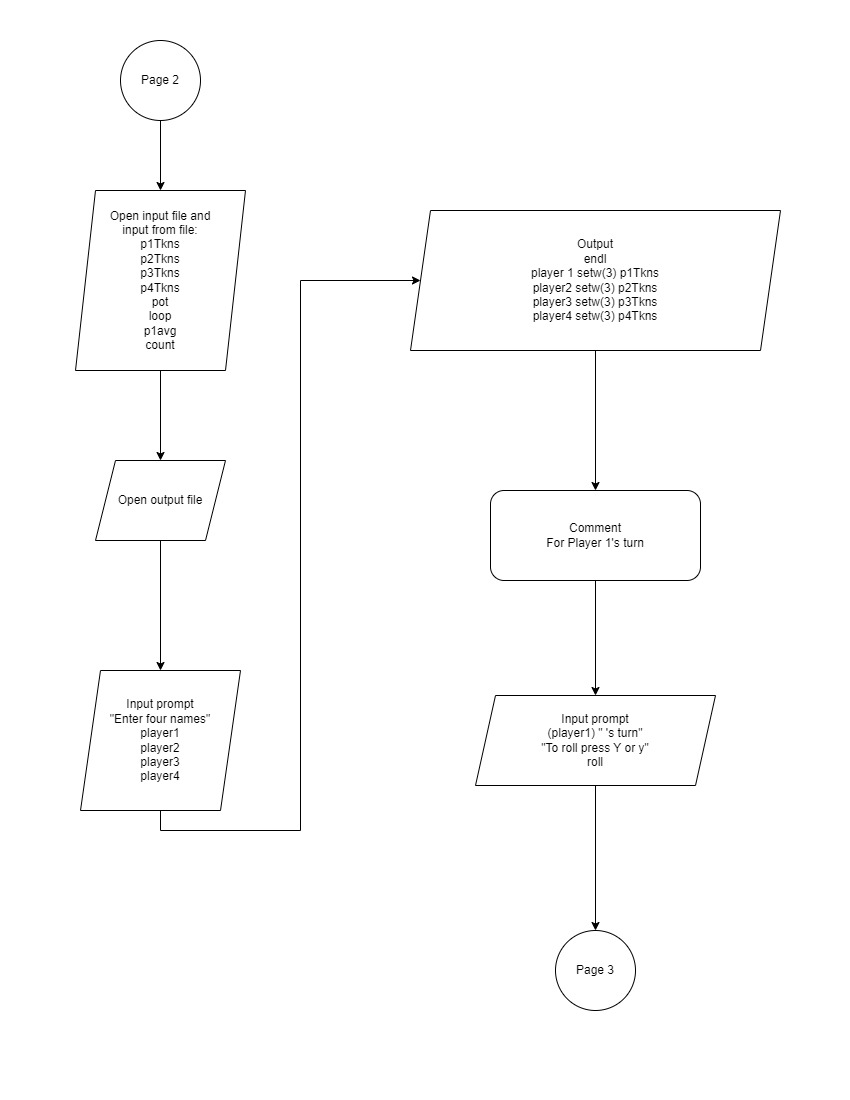
-Final version with a cleaner program, satisfying more Project II requirements, and fixing of previous errors

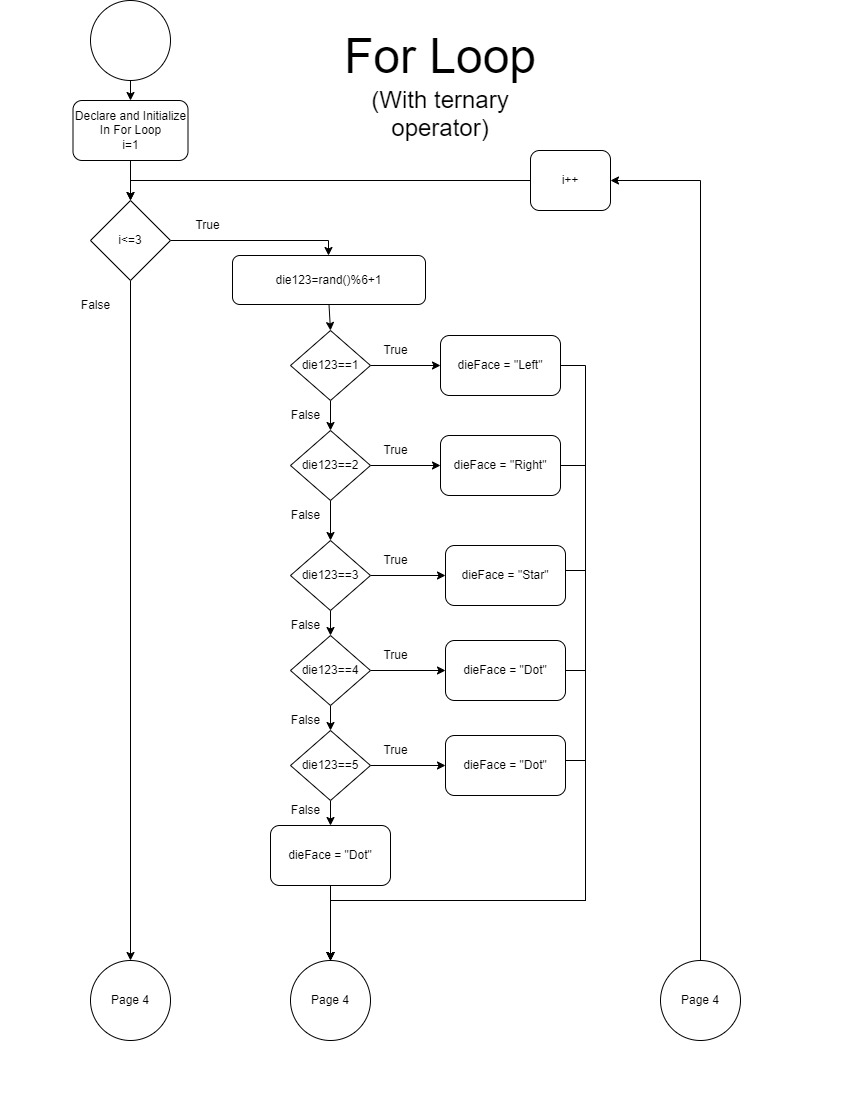
**Final Input From File**

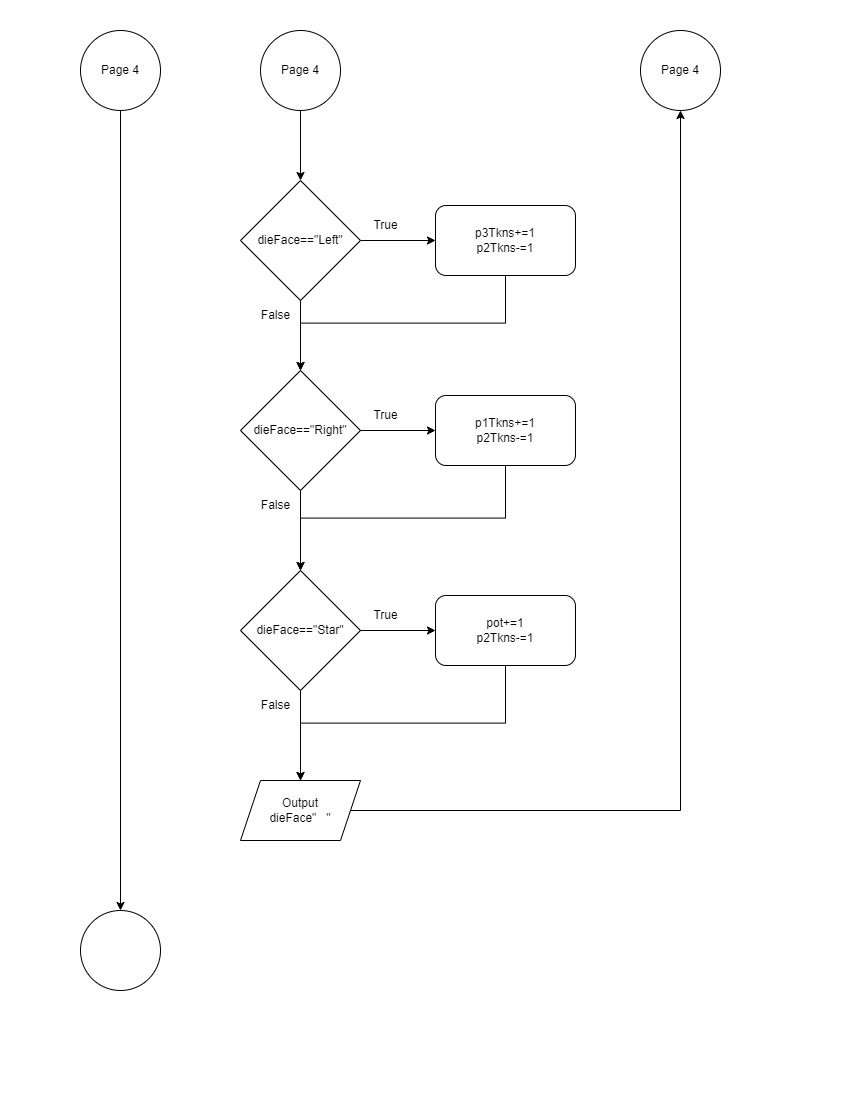
****

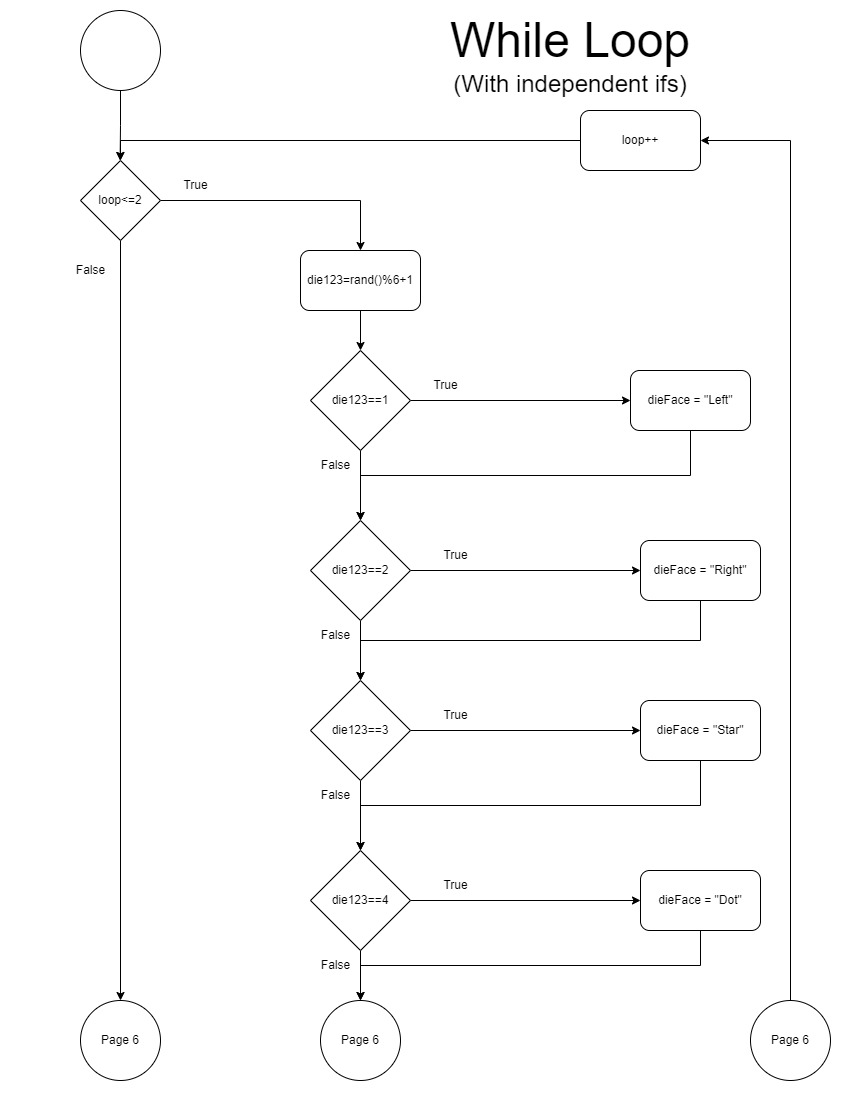
**-**Finalized inputs, adding in initialization of all 4 player averages.

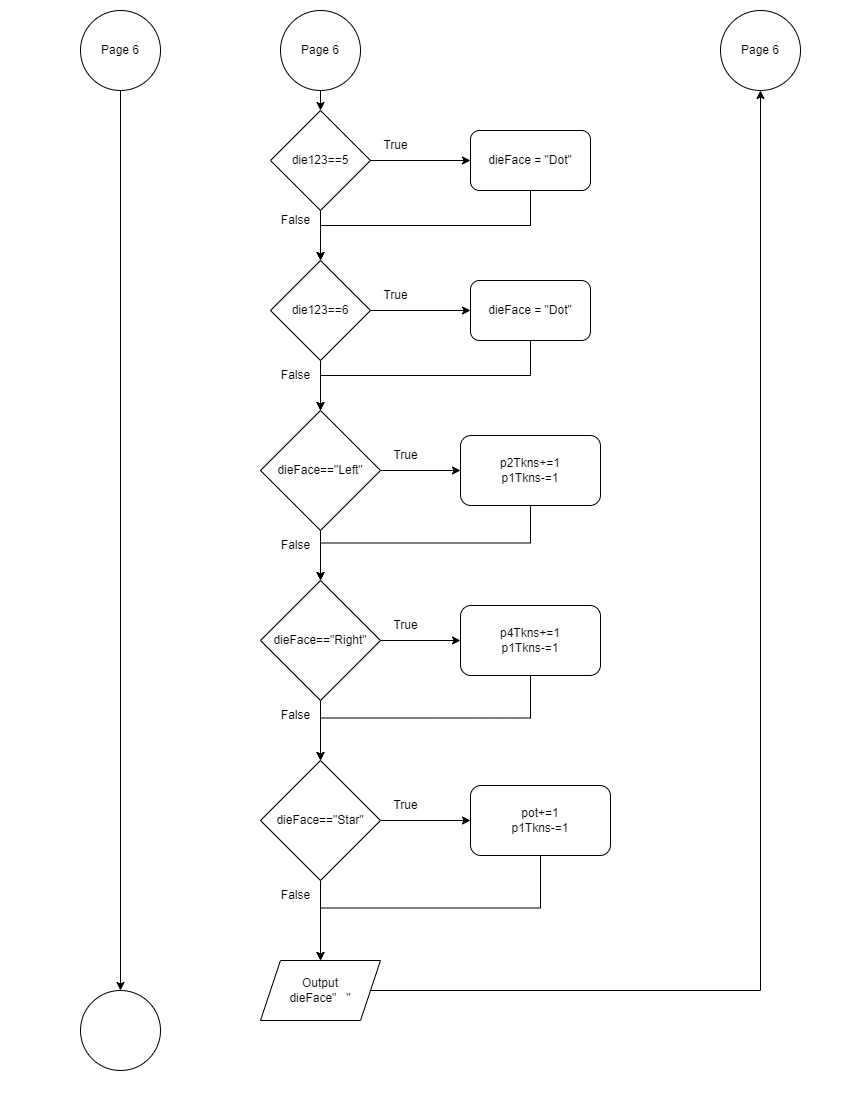
**   
Flowcharts**

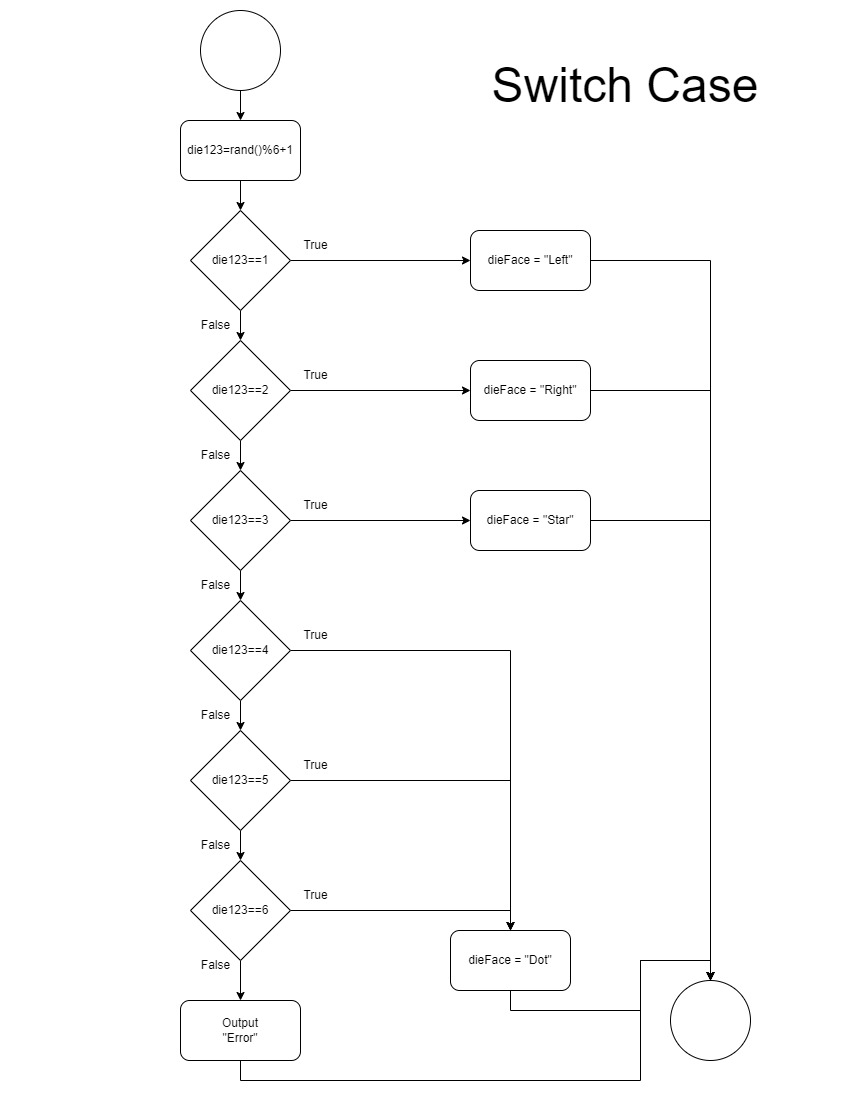
****

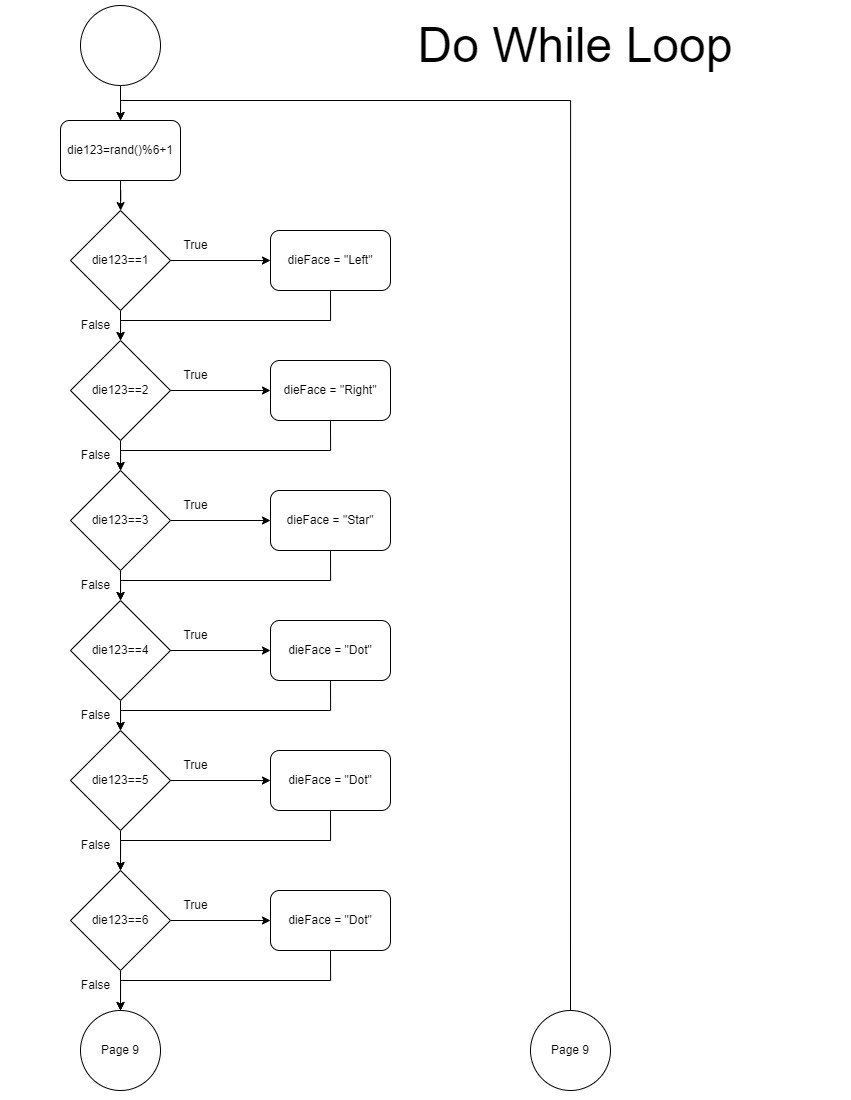
****

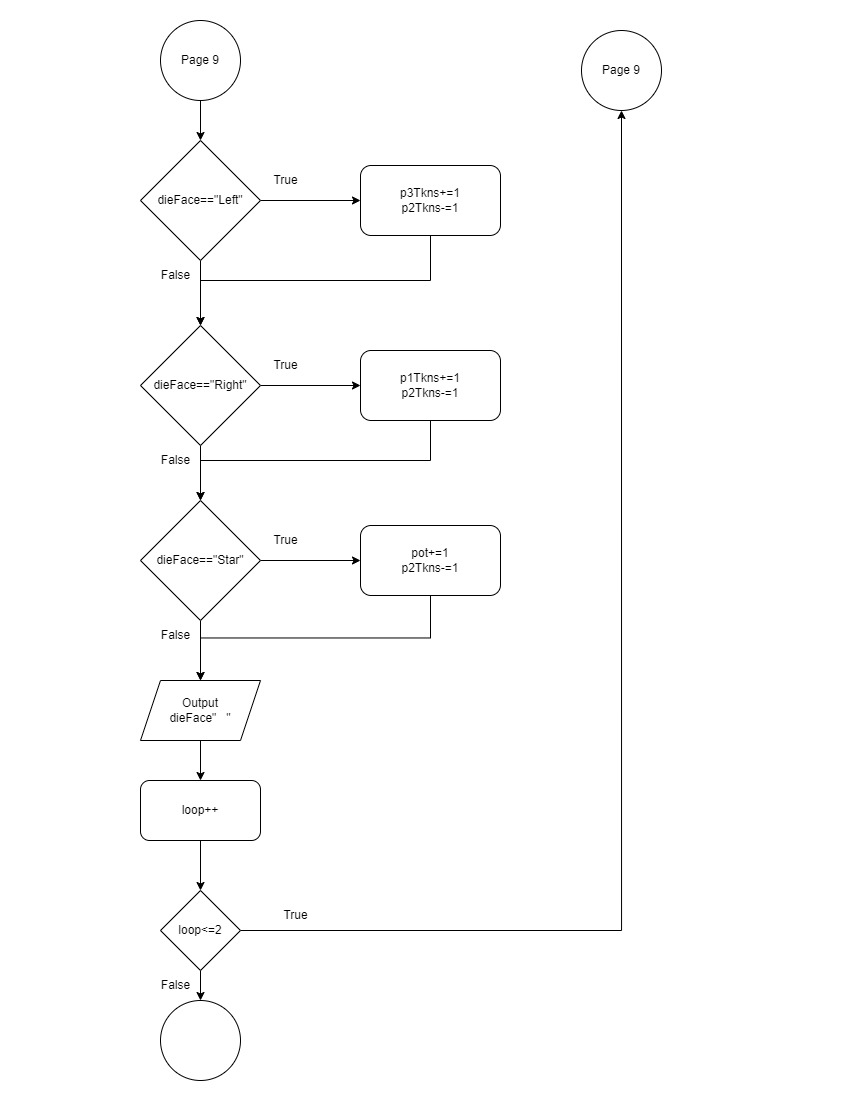
****

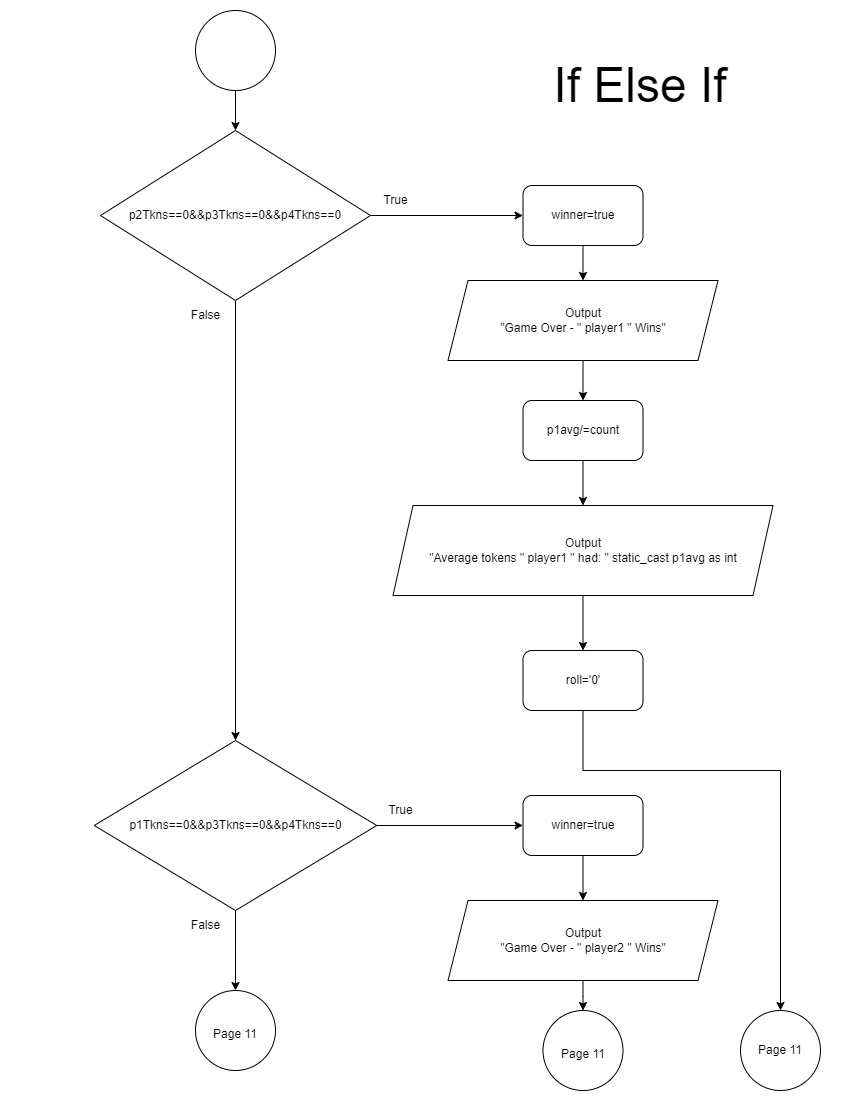


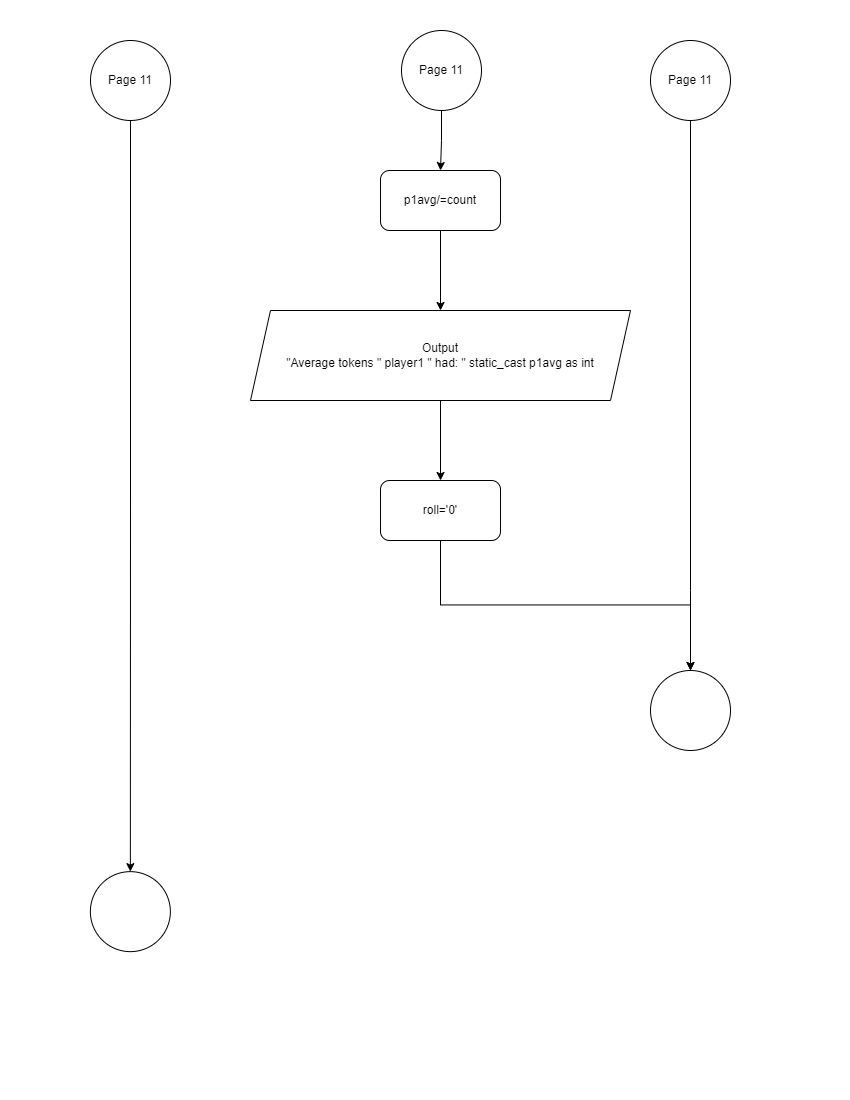


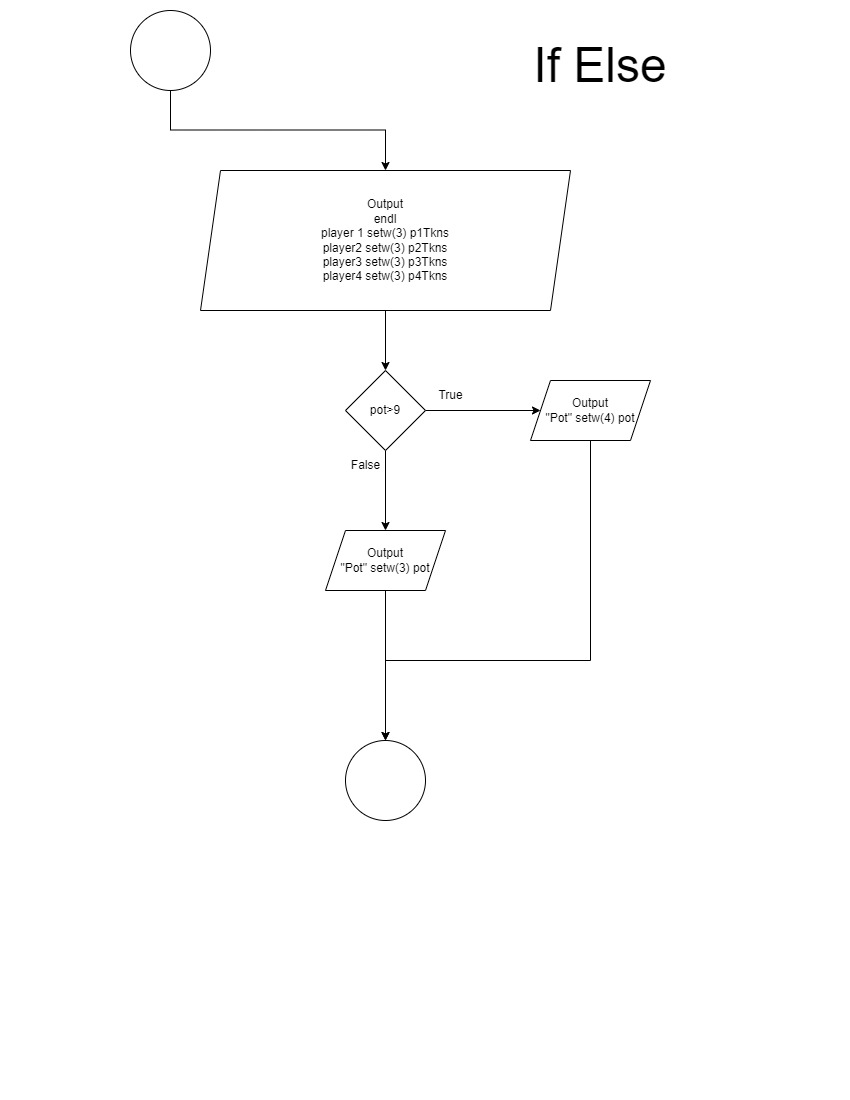


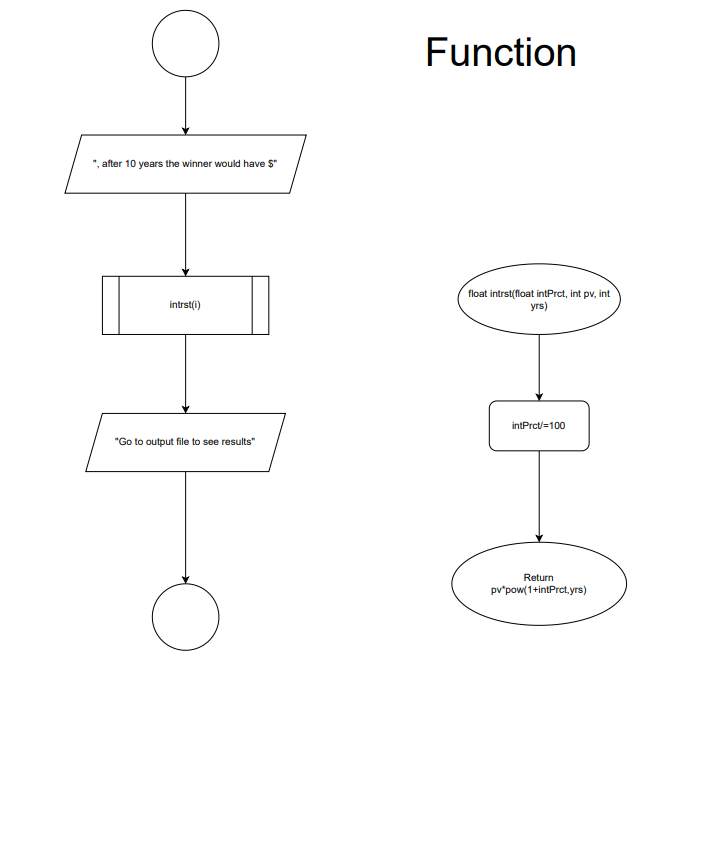












**Pseudo Code**

Intro message and how program works

Open I/O files

Read inputs from file

Input player names

Display tokens

Player 1 turn, input Y/y to roll

Outer loop start (do-while)

If Y/y, “roll” dice depending on # of tokens

Increase/decrease tokens function

Display results

If 0 tokens, skip turn

Output tokens

Put Player 1 tokens in array function

Add player tokens for player token average

Increase count (how many sets of tokens)

If only one player has tokens, player wins, calculate Player 1’s average tokens, set roll equal to 0 and program ends

Else, Player 2 turn, input Y/y to roll

If Y/y, “roll” dice depending on # of tokens

Increase/decrease tokens function

Display results

If 0 tokens, skip turn

Output tokens

Put Player 1 tokens in array function

Add Player 1’s tokens (used to calculate average tokens P1 had)

Increase count (how many sets of tokens)

If only one player has tokens, player wins, calculate player 1’s average tokens, set roll equal to 0 and program ends

Else, Player 3 turn, input Y/y to roll

If Y/y, “roll” dice depending on # of tokens

Increase/decrease tokens function

Display results

If 0 tokens, skip turn

Output tokens

Put Player 1 tokens in array function

Add Player 1’s tokens (used to calculate average tokens P1 had)

Increase count (how many sets of tokens)

If only one player has tokens, player wins, calculate Player 1’s average tokens, set roll equal to 0 and program ends

Else, Player 4 turn, input Y/y to roll

If Y/y, “roll” dice depending on # of tokens

Increase/decrease tokens function

Display results

If 0 tokens, skip turn

Output tokens

Put Player 1 tokens in array function

Add Player 1’s tokens (used to calculate average tokens P1 had)

Increase count (how many sets of tokens)

If only one player has tokens, player wins, calculate Player 1’s average tokens, set roll equal to 0 and program ends

Else, Player 1 turn, input Y/y to roll

Outer loop ends when roll not equal to Y/y

Display output messages

Implement functions

Return results

Close input and output files

Close program

Function p1turn

If Left, increase/decrease specified tokens

If Right, increase/decrease specified tokens

If Star, increase/decrease specified tokens

(Same for function p2 turn, p3 turn, p4 turn)

Function intrst

Convert percent to decimal

Return calculation of future value

Function statTbl

Set array p1Tkns value

Set parallel arrays sumTkns and avgTkns

Function statTbl

Display statistic table

Function bublSrt

Initialize bool

Do while swap

If next average array position is smaller, swap

Function p1ary

Set array values for tokens Player 1 had each round

Function selSrt

If next Player 1 token array position is smaller, swap

Function linSrch

Find max value in Player 1 token array

Function end

Display final message

Exit function

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cross Reference for Project 1** | | | | | | | | | | | |
|  | |  | |  | |  | | |  |  | |
| **You are to fill-in with where located in**  **code** | | | | | | | | |  |  | |
|  | |  | |  | |  | |  |  | | |
|  | |  | |  | |  | |  |  | | |
| **Chapter** | | **Section** | | **Topic** | | **Where Line #''s** | | **Pts** | **Notes** | | |
| 2 | | 2 | | cout | |  | |  |  | | |
|  | | 3 | | libraries | | Lines 9-15 | | 5 | iostream, iomanip, cmath, cstdlib, fstream, string, ctime | | |
|  | | 4 | | variables/literals | |  | |  | No variables in global area, failed project! | | |
|  | | 5 | | Identifiers | |  | |  |  | | |
|  | | 6 | | Integers | | Lines 49-61 | | 1 |  | | |
|  | | 7 | | Characters | | Line 62 | | 1 |  | | |
|  | | 8 | | Strings | | Lines 43-47 | | 1 |  | | |
|  | | 9 | | Floats No Doubles | | Line 63-68 | | 1 | Using doubles will fail the project, floats OK! | | |
|  | | 10 | | Bools | | Line 192 | | 1 |  | | |
|  | | 11 | | Sizeof \*\*\*\*\* | |  | |  |  | | |
|  | | 12 | | Variables 7 characters or less | |  | |  | All variables <= 7 characters | | |
|  | | 13 | | Scope \*\*\*\*\* No Global Variables | |  | |  |  | | |
|  | | 14 | | Arithmetic operators | |  | |  |  | | |
|  | | 15 | | Comments 20%+ | | Lines 42-70 and 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 | | Line 566 | | 1 |  | | |
|  | | 6 | | Multiple assignment \*\*\*\*\* | |  | |  |  | | |
|  | | 7 | | Formatting output | | Lines 171-179 | | 1 |  | | |
|  | | 8 | | Strings | | Lines 224-228 | | 1 |  | | |
|  | | 9 | | Math Library | | Line 648 | | 1 | All libraries included have to be used | | |
|  | | 10 | | Hand tracing \*\*\*\*\*\* | |  | |  |  | | |
|  | |  | |  | |  | |  |  | | |
| 4 | | 1 | | Relational Operators | |  | |  |  | | |
|  | | 2 | | if | | Lines 239-250 | | 1 | Independent if | | |
|  | | 4 | | If-else | | Lines 285-289 | | 1 |  | | |
|  | | 5 | | Nesting | | Lines 330-388 | | 1 |  | | |
|  | | 6 | | If-else-if | | Lines 412-428 | | 1 |  | | |
|  | | 7 | | Flags \*\*\*\*\* | |  | |  |  | | |
|  | | 8 | | Logical operators | | Lines 424, 429, and 438 | | 1 |  | | |
|  | | 11 | | Validating user input | | Lines 109 and 543 | | 1 |  | | |
|  | | 13 | | Conditional Operator | | Lines 443-447 | | 1 |  | | |
|  | | 14 | | Switch | | Lines 481-489 | | 1 |  | | |
|  | |  | |  | |  | |  |  | | |
| 5 | | 1 | | Increment/Decrement | | Lines 642 and 660 | | 1 |  | | |
|  | | 2 | | While | | Lines 128-141 | | 1 |  | | |
|  | | 5 | | Do-while | | Lines 237-257 | | 1 |  | | |
|  | | 6 | | For loop | | Lines 222-234 | | 1 |  | | |
|  | | 11 | | Files input/output both | | Lines 88-89 and 585-586 | | 2 |  | | |
|  | | 12 | | No breaks in loops \*\*\*\*\*\* | |  | |  | Failed Project if included | | |
|  | |  | |  | |  | |  |  | | |
|  | |  | |  | |  | |  |  | | |
| \*\*\*\*\*\* Not required to show | |  | |  | | Total | | 30 |  | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cross Reference for Project 2** | | | | | |
|  |  |  |  |  |  |
| **You are to fill-in with where located in code** | | | |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 6 |  | Functions |  |  |  |
|  | 3 | Function Prototypes | Lines 24-35 | 4 | Always use prototypes |
|  | 5 | Pass by Value | Lines 73,559, and 646 | 4 |  |
|  | 8 | return | Line 648 | 4 | A value from a function |
|  | 9 | returning boolean |  | 4 |  |
|  | 10 | Global variables |  | XXX | Do not use global variables -100 pts |
|  | 11 | Static variables | Line 706 | 4 |  |
|  | 12 | defaulted arguments | Line 28 | 4 |  |
|  | 13 | pass by reference | Line 594 | 4 |  |
|  | 14 | overloading | Lines 652&666 | 5 |  |
|  | 15 | exit() function | Line 737 | 4 |  |
| 7 |  | Arrays |  |  |  |
|  | 1 to 6 | Single Dimensioned Arrays | Lines 655-658 | 3 |  |
|  | 7 | Parallel Arrays | Lines 660-663 | 2 |  |
|  | 8 | Single Dimensioned as Function Arguments | Line 652 | 2 |  |
|  | 9 | 2 Dimensioned Arrays |  | 2 | Emulate style in book/in class repository |
|  | 12 | STL Vectors |  | 2 |  |
|  |  | Passing Arrays to and from Functions | Lines 705, 712, and 575-576 | 5 |  |
|  |  | Passing Vectors to and from Functions |  | 5 |  |
|  |  |  |  |  |  |
|  |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort | Lines 689-702 | 4 |  |
|  | 3 | Selection Sort | Lines 712-722 | 4 |  |
|  | 1 | Linear or Binary Search | Lines 725-732 | 4 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not required to show |  |  | Total | 70 | Other 30 points from Proj 1 first sheet tab |

**Program**

/\*

\* File: main.cpp

\* Author: Jazmyne Patlan

\* Created on February 11th, 2023, 6:24 PM

\* Purpose: Game of Left Center Right

\*/

//System Libraries

#include <iostream> //Input Output Library

#include <string> //String Objects

#include <cstdlib> //Random Number Library

#include <ctime> //Time Library

#include <iomanip> //Format Library

#include <fstream> //Read and Write to a File

#include <cmath> //Math Library

using namespace std;

//User Libraries

//Global Constants not Variables

//Science, Math, Conversions, Dimensions

//Function Prototypes

void p1turn(int &, int &, int &, int &, string &); //Increase/decrease tokens for player 1's turn

void p2turn(int &, int &, int &, int &, string &); //Increase/decrease tokens for player 2's turn

void p3turn(int &, int &, int &, int &, string &); //Increase/decrease tokens for player 3's turn

void p4turn(int &, int &, int &, int &, string &); //Increase/decrease tokens for player 4's turn

float intrst(float, int=3, int=10); //Calculate interest

void statTbl(float, int [], int [], float [], int , int, int, int); //Set arrays for table with token sums, sets, and averages

void statTbl(float, int [], int [], float [], string, string, string, string); //Display table of token sums, sets, and averages

void bublSrt(float []); //Sort average tokens array

void p1ary(int, int []); //Array for tokens player 1 had each set

void selSrt(int [],int); //Sort array of tokens player 1 had each set

int linSrch(int, int[], int); //Search for max value of tokens player 1 had in a single set

void end(); //End program

//Execution begins here at main

int main(int argc, char\*\* argv) {

//Set random number seed

srand(static\_cast<unsigned int>(time(0)));

//Declare Variables

string player1; //Player 1's name

string player2; //Player 2's name

string player3; //Player 3's name

string player4; //Player 4's name

string dieFace; //What each side of the die reads

const int SIZE=500; //Constant max size for arrays

int p1Tkns, //Player 1's tokens

p2Tkns, //Player 2's tokens

p3Tkns, //Player 3's tokens

p4Tkns, //Player 4's tokens

die123, //Random value for each die roll

pot, //Pot where tokens go if a Star is rolled

loop, //Determines where while loop stops

count, //Counts how many sets of tokens there are

max, //Max value in player 1 token array

indx, //Index marks position in array where a value is

sumTkns[SIZE], //Array for individual sum of tokens for each player

plrTkns[SIZE], //Array for looped sum of tokens for each player

p1array[SIZE]; //Array for tokens player 1 had each set

char roll; //Determines if a calculation of a roll should occur

float i, //Interest in percent form

p1avg, //Average tokens player 1 had

p2avg, //Average tokens player 2 had

p3avg, //Average tokens player 3 had

p4avg, //Average tokens player 4 had

avgTkns[SIZE]; //Array for average tokens players had

fstream in, //Read in initialization for variables

out; //Write out results of game and winnings

//Initialize Variables

i=6; //Percent interest

indx=0; //Position where max is found

p1array[0]=3; //First value in player 1 tokens array

//Intro message

cout<<"Game of Left Center Right"<<endl;

cout<<"This program requires four players"<<endl;

cout<<"Players are arranged in a circle facing the Pot, as displayed";

cout<<" below:"<<endl<<endl;

//Display player orientation that program works for

cout<<" Player 3 "<<endl;

cout<<"Player 2 Pot Player 4"<<endl;

cout<<" Player 1 "<<endl<<endl;

in.open("input7.dat",ios::in); //Input and output files

out.open("LCRoutput7.dat",ios::out);

in>>p1Tkns>>p2Tkns>>p3Tkns>>p4Tkns>>pot>>loop>>p1avg>>p2avg>>p3avg>>p4avg>>count;

//Input prompt for player names

cout<<"Enter four names"<<endl;

cin>>player1>>player2>>player3>>player4;

cout<<endl;

//Map/Process the Inputs -> Outputs

cout<<"Each player starts with three tokens and the Pot starts empty"<<endl;

cout<<player1<<setw(3)<<p1Tkns<<endl; //Output player and tokens

cout<<player2<<setw(3)<<p2Tkns<<endl;

cout<<player3<<setw(3)<<p3Tkns<<endl;

cout<<player4<<setw(3)<<p4Tkns<<endl;

cout<<"Pot"<<setw(3)<<pot<<endl<<endl; //Output pot tokens

//For Player 1's turn

cout<<player1<<"'s turn"<<endl;

cout<<"To roll press Y or y"<<endl;

cin>>roll;

do{ //Continues program as long as roll=='Y' or roll=='y'

if (roll=='Y'||roll=='y'){

if (p1Tkns>=3){ //If player has over 3 tokens, roll three dice

cout<<"Roll: ";

for (int i=1; i<=3; i++){

die123=rand()%6+1; //Sets random number between 1-6

dieFace = (die123==1)? "Left": //Find what each die reads

(die123==2)? "Right":

(die123==3)? "Star":

(die123==4)? "Dot":

(die123==5)? "Dot": "Dot";

//Increase/decrease tokens for player 1 turn

p1turn(p1Tkns, p2Tkns, p4Tkns, pot, dieFace);

cout<<dieFace<<" "; //Display dice results

}

}else if (p1Tkns==2){

cout<<"Roll: ";

while (loop<=2){ //If player has 2 tokens, loop die roll twice

die123=rand()%6+1; //Sets random number between 1-6

if (die123==1){

dieFace="Left";} //Die reads left if random number is 1

if (die123==2){

dieFace="Right";}//Die reads right if random number is 2

if (die123==3){

dieFace="Star";} //Die reads star if random number is 3

if (die123==4){

dieFace="Dot";} //Die reads dot for other random numbers

if (die123==5){

dieFace="Dot";}

if (die123==6){

dieFace="Dot";}

//Increase/decrease tokens for player 1 turn

p1turn(p1Tkns, p2Tkns, p4Tkns, pot, dieFace);

cout<<dieFace<<" "; //Display dice results

loop++; //Increment loop until at 2

}

loop=1; //Set loop back to 1 so can be used again

}else if (p1Tkns==1){ //If player has 1 token, roll die once

cout<<"Roll: ";

die123=rand()%6+1; //Sets random number between 1-6

switch(die123){

case 1: dieFace="Left";break; //Die reads left if 1

case 2: dieFace="Right";break; //Die reads right if 2

case 3: dieFace="Star";break; //Die reads star if 1

case 4: //Die reads dot if 4, 5, or 6

case 5:

case 6: dieFace="Dot";break;

default:cout<<"Error"<<endl; //If any other number, error

}

//Increase/decrease tokens for player 1 turn

p1turn(p1Tkns, p2Tkns, p4Tkns, pot, dieFace);

cout<<dieFace<<" "; //Display dice results

}else{ //If zero tokens, skip turn

cout<<"Skip turn, same amount of tokens remain";

}

cout<<endl;

cout<<player1<<setw(3)<<p1Tkns<<endl; //Display each players tokens

cout<<player2<<setw(3)<<p2Tkns<<endl;

cout<<player3<<setw(3)<<p3Tkns<<endl;

cout<<player4<<setw(3)<<p4Tkns<<endl;

if (pot>9){ //Set width based on tokens in pot

cout<<"Pot"<<setw(4)<<pot<<endl<<endl;

}else{

cout<<"Pot"<<setw(3)<<pot<<endl<<endl;

}

p1ary(p1Tkns,p1array); //Put player 1's tokens in an array

p1avg+=p1Tkns; //Add players tokens together for average calculation

p2avg+=p2Tkns;

p3avg+=p3Tkns;

p4avg+=p4Tkns;

count++; //Count each set of tokens

bool winner=false;

if (p2Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player1<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player2<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player3<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p3Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player4<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}

if (!winner){ //If no players won, display input message

cout<<player2<<"'s turn"<<endl;

cout<<"To roll press Y or y"<<endl;

cin>>roll;

}

}

//For Player 2's turn

if (roll=='Y'||roll=='y'){

if (p2Tkns>=3){ //If player has over 3 tokens, roll three dice

cout<<"Roll: ";

for (int i=1; i<=3; i++){

die123=rand()%6+1; //Sets random number between 1-6

dieFace = (die123==1)? "Left": //Find what each die reads

(die123==2)? "Right":

(die123==3)? "Star":

(die123==4)? "Dot":

(die123==5)? "Dot": "Dot";

//Increase/decrease tokens for player 2 turn

p2turn(p1Tkns, p2Tkns, p3Tkns, pot, dieFace);

cout<<dieFace<<" "; //Display dice results

}

}else if (p2Tkns==2){ //If player has 2 tokens, roll two dice

cout<<"Roll: ";

do{ //Set one die result before test

die123=rand()%6+1;

if (die123==1){

dieFace="Left";}//Die reads left if random number is 1

if (die123==2){

dieFace="Right";}//Die reads right if random number is 2

if (die123==3){

dieFace="Star";}//Die reads star if random number is 3

if (die123==4){

dieFace="Dot";} //Die reads dot for other random numbers

if (die123==5){

dieFace="Dot";}

if (die123==6){

dieFace="Dot";}

//Increase/decrease tokens for player 2 turn

p2turn(p1Tkns, p2Tkns, p3Tkns, pot, dieFace);

cout<<dieFace<<" "; //Display dice results

loop++; //Increment loop until at 2

}while (loop<=2);

loop=1; //Set loop back to 1 so can be used again

}else if (p2Tkns==1){ //If player has 1 token, roll die once

cout<<"Roll: ";

die123=rand()%6+1; //Sets random number between 1-6

switch(die123){

case 1: dieFace="Left";break; //Die reads left if 1

case 2: dieFace="Right";break; //Die reads right if 2

case 3: dieFace="Star";break; //Die reads star if 1

case 4: //Die reads dot if 4, 5, or 6

case 5:

case 6: dieFace="Dot";break;

default:cout<<"Error"<<endl; //If any other number, error

}

//Increase/decrease tokens for player 2 turn

p2turn(p1Tkns, p2Tkns, p3Tkns, pot, dieFace);

cout<<dieFace<<" "; //Display dice results

}else{ //If zero tokens, skip turn

cout<<"Skip turn, same amount of tokens remain";

}

cout<<endl;

cout<<player1<<setw(3)<<p1Tkns<<endl; //Display each players tokens

cout<<player2<<setw(3)<<p2Tkns<<endl;

cout<<player3<<setw(3)<<p3Tkns<<endl;

cout<<player4<<setw(3)<<p4Tkns<<endl;

if (pot>9){ //Set width based on tokens in pot

cout<<"Pot"<<setw(4)<<pot<<endl<<endl;

}else{

cout<<"Pot"<<setw(3)<<pot<<endl<<endl;

}

p1ary(p1Tkns,p1array); //Put player 1's tokens in an array

p1avg+=p1Tkns; //Add players tokens together for average calculation

p2avg+=p2Tkns;

p3avg+=p3Tkns;

p4avg+=p4Tkns;

count++; //Count each set of tokens

bool winner=false;

if (p2Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player1<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player2<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player3<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p3Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player4<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}

if (!winner){ //If no players won, display input message

cout<<player3<<"'s turn"<<endl;

cout<<"To roll press Y or y"<<endl;

cin>>roll;

}

}

//For Player 3's turn

if (roll=='Y'||roll=='y'){

if (p3Tkns>=3){ //If player has over 3 tokens, roll three dice

cout<<"Roll: ";

for (int i=1; i<=3; i++){

die123=rand()%6+1; //Sets random number between 1-6

dieFace = (die123==1)? "Left": //Find what each die reads

(die123==2)? "Right":

(die123==3)? "Star":

(die123==4)? "Dot":

(die123==5)? "Dot": "Dot";

//Increase/decrease tokens for player 3 turn

p3turn(p2Tkns,p3Tkns,p4Tkns,pot,dieFace);

cout<<dieFace<<" "; //Display dice results

}

}else if (p3Tkns==2){

cout<<"Roll: ";

while (loop<=2){ //If player has 2 tokens, loop die roll twice

die123=rand()%6+1; //Sets random number between 1-6

if (die123==1){

dieFace="Left";} //Die reads left if random number is 1

if (die123==2){

dieFace="Right";}//Die reads right if random number is 2

if (die123==3){

dieFace="Star";} //Die reads star if random number is 3

if (die123==4){

dieFace="Dot";} //Die reads dot for other random numbers

if (die123==5){

dieFace="Dot";}

if (die123==6){

dieFace="Dot";}

//Increase/decrease tokens for player 3 turn

p3turn(p2Tkns,p3Tkns,p4Tkns,pot,dieFace);

cout<<dieFace<<" "; //Display dice results

loop++; //Increment loop until at 2

}

loop=1; //Set loop back to 1 so can be used again

}else if (p3Tkns==1){ //If player has 1 token, roll die once

cout<<"Roll: ";

die123=rand()%6+1; //Sets random number between 1-6

switch(die123){

case 1: dieFace="Left";break; //Die reads left if 1

case 2: dieFace="Right";break; //Die reads right if 2

case 3: dieFace="Star";break; //Die reads star if 1

case 4: //Die reads dot if 4, 5, or 6

case 5:

case 6: dieFace="Dot";break;

default:cout<<"Error"<<endl; //If any other number, error

}

//Increase/decrease tokens for player 3 turn

p3turn(p2Tkns, p3Tkns,p4Tkns,pot,dieFace);

cout<<dieFace<<" "; //Display dice results

}else{ //If zero tokens, skip turn

cout<<"Skip turn, same amount of tokens remain";

}

cout<<endl;

cout<<player1<<setw(3)<<p1Tkns<<endl; //Display each players tokens

cout<<player2<<setw(3)<<p2Tkns<<endl;

cout<<player3<<setw(3)<<p3Tkns<<endl;

cout<<player4<<setw(3)<<p4Tkns<<endl;

if (pot>9){ //Set width based on tokens in pot

cout<<"Pot"<<setw(4)<<pot<<endl<<endl;

}else{

cout<<"Pot"<<setw(3)<<pot<<endl<<endl;

}

p1ary(p1Tkns,p1array); //Put player 1's tokens in an array

p1avg+=p1Tkns; //Add players tokens together for average calculation

p2avg+=p2Tkns;

p3avg+=p3Tkns;

p4avg+=p4Tkns;

count++; //Count each set of tokens

bool winner=false;

if (p2Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player1<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player2<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player3<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p3Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player4<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}

if (!winner){ //If no players won, display input message

cout<<player4<<"'s turn"<<endl;

cout<<"To roll press Y or y"<<endl;

cin>>roll;

}

}

//For Player 4's turn

if (roll=='Y'||roll=='y'){

if (p4Tkns>=3){ //If player has over 3 tokens, roll three dice

cout<<"Roll: ";

for (int i=1; i<=3; i++){

die123=rand()%6+1; //Sets random number between 1-6

dieFace = (die123==1)? "Left": //Find what each die reads

(die123==2)? "Right":

(die123==3)? "Star":

(die123==4)? "Dot":

(die123==5)? "Dot": "Dot";

//Increase/decrease tokens for player 4 turn

p4turn(p1Tkns,p3Tkns,p4Tkns,pot,dieFace);

cout<<dieFace<<" "; //Display dice results

}

}else if (p4Tkns==2){

cout<<"Roll: ";

while (loop<=2){ //If player has 2 tokens, loop die roll twice

die123=rand()%6+1; //Sets random number between 1-6

if (die123==1){

dieFace="Left";} //Die reads left if random number is 1

if (die123==2){

dieFace="Right";}//Die reads right if random number is 2

if (die123==3){

dieFace="Star";} //Die reads star if random number is 3

if (die123==4){

dieFace="Dot";} //Die reads dot for other random numbers

if (die123==5){

dieFace="Dot";}

if (die123==6){

dieFace="Dot";}

//Increase/decrease tokens for player 4 turn

p4turn(p1Tkns,p3Tkns,p4Tkns,pot,dieFace);

cout<<dieFace<<" "; //Display dice results

loop++; //Increment loop until at 2

}

loop=1; //Set loop back to 1 so can be used again

}else if (p4Tkns==1){ //If player has 1 token, roll die once

cout<<"Roll: ";

die123=rand()%6+1; //Sets random number between 1-6

switch(die123){

case 1: dieFace="Left";break; //Die reads left if 1

case 2: dieFace="Right";break; //Die reads right if 2

case 3: dieFace="Star";break; //Die reads star if 1

case 4: //Die reads dot if 4, 5, or 6

case 5:

case 6: dieFace="Dot";break;

default:cout<<"Error"<<endl; //If any other number, error

}

//Increase/decrease tokens for player 4 turn

p4turn(p1Tkns,p3Tkns,p4Tkns,pot,dieFace);

cout<<dieFace<<" "; //Display dice results

}else{ //If zero tokens, skip turn

cout<<"Skip turn, same amount of tokens remain";

}

cout<<endl;

cout<<player1<<setw(3)<<p1Tkns<<endl; //Display each players tokens

cout<<player2<<setw(3)<<p2Tkns<<endl;

cout<<player3<<setw(3)<<p3Tkns<<endl;

cout<<player4<<setw(3)<<p4Tkns<<endl;

if (pot>9){ //Set width based on tokens in pot

cout<<"Pot"<<setw(4)<<pot<<endl<<endl;

}else{

cout<<"Pot"<<setw(3)<<pot<<endl<<endl;

}

p1ary(p1Tkns,p1array); //Put player 1's tokens in an array

p1avg+=p1Tkns; //Add players tokens together for average calculation

p2avg+=p2Tkns;

p3avg+=p3Tkns;

p4avg+=p4Tkns;

count++; //Count each set of tokens

bool winner=false;

if (p2Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player1<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p3Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player2<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p4Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player3<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}else if (p1Tkns==0&&p2Tkns==0&&p3Tkns==0){ //If one player has...

winner=true; //tokens they win

out<<"GAME OVER - "<<player4<<" Wins"<<endl;

roll='0'; //Set roll to zero to exit loop

}

if (!winner){ //If no players won, display input message

cout<<player1<<"'s turn"<<endl;

cout<<"To roll press Y or y"<<endl;

cin>>roll;

}

}

}while (roll=='Y'||roll=='y'); //Loop back to player 1's turn

//Display Inputs/Outputs

//Output file messages

out<<endl<<endl<<"One version of the game includes substituting the tokens";

out<<" for quarters"<<endl;

out<<"The winner is then entitled to all quarters, even the ones";

out<<" in the Pot"<<endl;

//Display potential wins

out<<"If players were gambling with this program, the winner would be";

out<<" entitled to $3"<<endl;

//Output money after interest

out<<"If the winner were to invest the $3 dollars with 6% interest";

out<<", after 10 years the winner would have $";

out<<setprecision(2)<<fixed<<intrst(i)<<endl;

//Message directing to output file and statistic table

cout<<"Go to output file to see results"<<endl<<endl<<endl;

cout<<"Look below to see some bonus game statistics:"<<endl<<endl;

//Initialize arrays and display table

statTbl(static\_cast<float>(count), plrTkns, sumTkns, avgTkns, p1avg, p2avg,

p3avg, p4avg);

statTbl(static\_cast<float>(count), plrTkns, sumTkns, avgTkns, player1,

player2,player3,player4);

//Sort function for token averages

bublSrt(avgTkns);

cout<<"The lowest token average was: "<<avgTkns[0]<<endl;

selSrt(p1array,count); //Sort functions for player 1 tokens each set

max=p1array[count-1]; //Set max value

indx=linSrch(count, p1array, max); //Linear search function to find array position

if (indx!=-1){ //If max value was found, display it and it's position

cout<<"The max amount of tokens "<<player1<<" had during a round was: ";

cout<<max<<", which was found at index = "<<indx<<endl<<endl;

}

//Clean up memory and files

in.close();

out.close();

end(); //Function to display final message and exit

//Exit the Program

return 0;

}

//Increase/decrease tokens for player 1's turn

void p1turn(int &p1Tkns, int &p2Tkns, int &p4Tkns, int &pot, string &dieFace){

if (dieFace=="Left"){ //Increase/decrease tokens for left

p2Tkns+=1;

--p1Tkns;}

if (dieFace=="Right"){ //Increase/decrease tokens for right

p4Tkns+=1;

--p1Tkns;}

if (dieFace=="Star"){ //Increase/decrease tokens for star

pot+=1;

--p1Tkns;}

}

//Increase/decrease tokens for player 2's turn

void p2turn(int &p1Tkns, int &p2Tkns, int &p3Tkns, int &pot, string &dieFace){

if (dieFace=="Left"){ //Increase/decrease tokens for left

p3Tkns+=1;

--p2Tkns;}

if (dieFace=="Right"){ //Increase/decrease tokens for right

p1Tkns+=1;

--p2Tkns;}

if (dieFace=="Star"){ //Increase/decrease tokens for star

pot+=1;

--p2Tkns;}

}

//Increase/decrease tokens for player 3's turn

void p3turn(int &p2Tkns, int &p3Tkns, int &p4Tkns, int &pot, string &dieFace){

if (dieFace=="Left"){ //Increase/decrease tokens for left

p4Tkns+=1;

--p3Tkns;}

if (dieFace=="Right"){ //Increase/decrease tokens for right

p2Tkns+=1;

--p3Tkns;}

if (dieFace=="Star"){ //Increase/decrease tokens for star

pot+=1;

--p3Tkns;}

}

//Increase/decrease tokens for player 4's turn

void p4turn(int &p1Tkns, int &p3Tkns, int &p4Tkns, int &pot, string &dieFace){

if (dieFace=="Left"){ //Increase/decrease tokens for left

p1Tkns+=1;

--p4Tkns;}

if (dieFace=="Right"){ //Increase/decrease tokens for right

p3Tkns+=1;

--p4Tkns;}

if (dieFace=="Star"){ //Increase/decrease tokens for star

pot+=1;

--p4Tkns;}

}

//Calculate interest for winnings

float intrst(float intPrct, int pv, int yrs){

intPrct/=100;

return pv\*pow(1+intPrct,yrs); //Equation to find future value

}

//Set arrays for statistic table

void statTbl(float count, int plrTkns[], int sumTkns[], float avgTkns[],

int p1avg, int p2avg, int p3avg, int p4avg){

plrTkns[0]=p1avg; //Set tokens array positions to total sum of each players tokens

plrTkns[1]=p2avg;

plrTkns[2]=p3avg;

plrTkns[3]=p4avg;

for (int j=0; j<4; j++){

sumTkns[j]= plrTkns[j]; //Set sum array to tokens array

avgTkns[j]=plrTkns[j]/count; //Set average array values from tokens array and count

}

}

//Display statistic table

void statTbl(float count, int plrTkns[], int sumTkns[], float avgTkns[],

string player1, string player2, string player3, string player4){

//Display table header

cout<<" Player Total Tokens # of Sets Average Tokens"<<endl;

for (int j=0; j<4; j++){

if (j==0){

cout<<setw(11)<<player1; //Output player name based on array position

}else if (j==1){

cout<<setw(11)<<player2;

}else if (j==2){

cout<<setw(11)<<player3;

}else if (j==3){ //For last position, display final name

cout<<setw(11)<<player4;

}

//Output sum of tokens, number of token sets, and average number of tokens per player

cout<<setw(15)<<sumTkns[j]<<setw(12)<<static\_cast<int>(count);

cout<<setw(17)<<fixed<<setprecision(2)<<avgTkns[j]<<endl;

}

cout<<endl;

}

//Sort average tokens array

void bublSrt(float avgTkns[]){

bool swap;

do{

swap=false; //Initialize bool to false

for(int i=0; i<3; i++){

if (avgTkns[i]>avgTkns[i+1]){ //If the value of the following array position is smaller, swap

float temp=avgTkns[i]; //Use temporary variable to complete swap

avgTkns[i]=avgTkns[i+1];

avgTkns[i+1]=temp;

swap=true; //Set swap to true if a swap occurred

}

}

}while(swap); //Swap as long as the swap bool is true

}

//Array for tokens player 1 had each set

void p1ary(int p1Tkns, int p1array[]){

static int k=1;

p1array[k]=p1Tkns; //Add token value into array

k++; //Increase array position every time function is called

}

//Sort array values for player 1 tokens

void selSrt(int p1array[],int count){

for (int k=0; k<count-1; k++){ //Loop for original positions

for (int j=k+1; j<count; j++){ //Loop for one position ahead

if(p1array[k]>p1array[j]){ //If position ahead is smaller, swap

int temp=p1array[k];

p1array[k]=p1array[j];

p1array[j]=temp; //Complete swap

}

}

}

}

//Search for max amount of tokens player 1 had in a set

int linSrch(int count, int p1array[], int max){

for (int k=0; k<count; k++){

if (p1array[k]==max){ //If the array position matches max value, return position

return k;

}

}

return -1; //If the value doesn't match any position, return -1

}

//Exit program

void end(){

cout<<"Goodbye"; //Final message

exit(0);

}