***YAHTZEE!***

Dice Game

Elise Chue

CSC 5

Spring 2018

42483

**TABLE OF CONTENTS**

**INTRODUCTION…………………………………………………………………………………...3**

**RULES………………………………………………………………………………………………3**

**DEVELOPMENT NOTES…………………………………………………………………………3**

**PSEUDOCODE…………………………………………………………………………………….4**

**FLOWCHART………………………………………………………………………………………5**

**CONSTRUCTS/CONCEPTS USED……………………………………………………………..8**

**SAMPLE OUTPUT………………………………………………………………………………..12**

**PROGRAM…………………………………………………………………………………………15**

**REFERENCES………………………………………………………………………………….....25**

**INTRODUCTION**

I was introduced to Yahtzee by my sister-in-law a few years ago. It’s a fun and simple game that has the thrill of competition and gambling (with statistics!) without having to go to a casino. It was known as the Yacht game since this was first played by a group of rich people on their yacht. The game didn’t gain much traction at first since it’s rules and general appeal did not catch on in the marketing as a “Poker Dice Game”. But then the company organized parties where the game was introduced to the guests and it’s been popular ever since.

It can be played alone or with as many people you’d like. It shares a few similarities with Poker in that scoring can be based off well known hand sequences (i.e. straight, full house, etc.) by rerolling for a more favorable result. I chose this game since I figured this game was straight forward, yet complex enough to challenge me.

**RULES**

* *Object of the Game*

Accumulate the most points.

* *Rules*

The game can be played from 1 to any number of players.

The object of Yahtzee is to obtain the highest score from throwing 5 dice.  
The game consists of 13 rounds. In each round, you roll the dice and then score the roll in one of 13 categories. You must score once in each category. The score is determined by a different rule for each category.  
The game ends once all 13 categories have been scored.

**DEVELOPMENT NOTES**

Actually programming a game has to be very literal which produces a little bit of a challenge -considering the amount of experience I have - when translating it to code. For instance, I would not be able to play against a computer if I wanted a multi-player game since the computer would need to have artificial intelligence to know which dice to keep, given their options.

I decided to opt for a full 2 player game which is a vast improvement from my Project 1 version where I only programmed three solo rounds. Using functions made it a lot easier since it was made in easily manageable chunks.

I left out the option to have “Upper Section” and Yahtzee *bonuses*. I also decided to not go by joker rules, but the user could easily bypass that if I’m being honest. Unfortunately, I did not include a verifying system to check results besides if they’ve chosen the score type already.

Given that this project requires a lot of things that weren’t necessarily vital to a working game, I used a handful of concepts that appear redundant (i.e. exit( ) and static variable) in order to prove I know how to use them.

**PSEUDOCODE**

Set random number seed

Declare Variables

Initialize Variables

Roll first Set for Player 1

Sort Dice for Player 1

Display Dice for Player 1

Reroll Dice for Player 1

Menu and Scoring for Player 1

Roll first Set for Player 2

Sort Player 2 Dice

Display Player 2 Dice

Reroll Player 2 Dice

Menu and Scoring for Player 2

Display Scoreboard

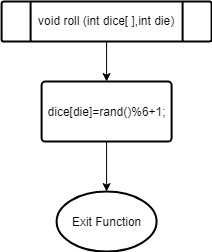
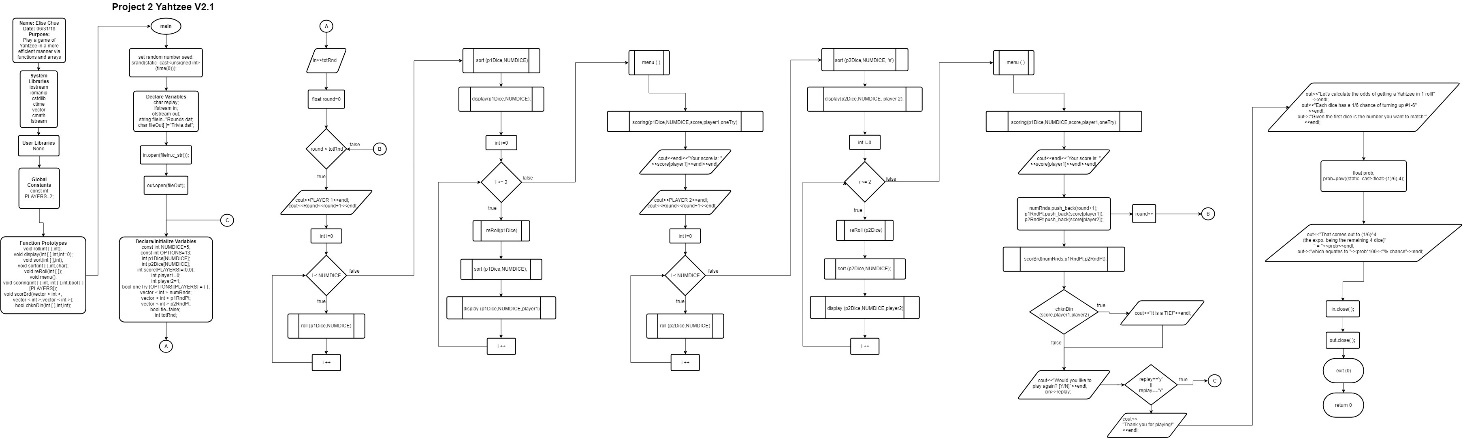
Ask Do you want to play again?

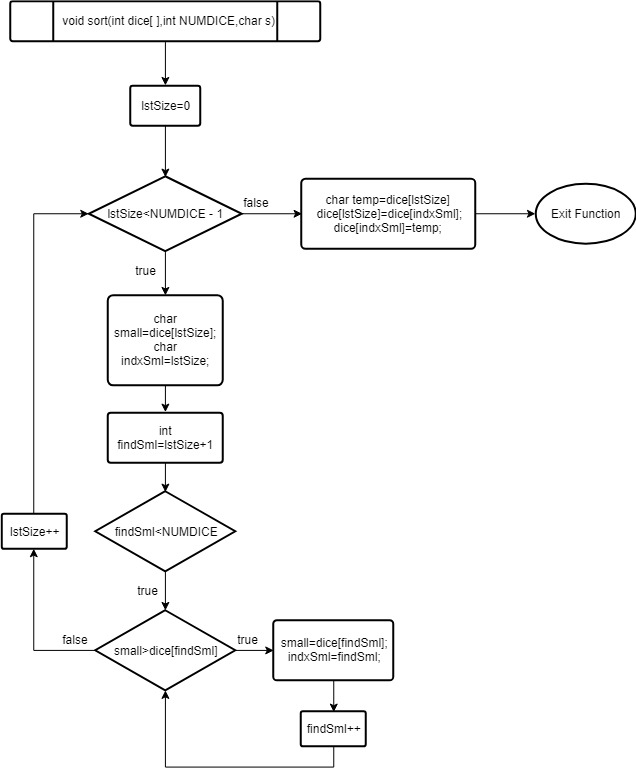
Exit Program

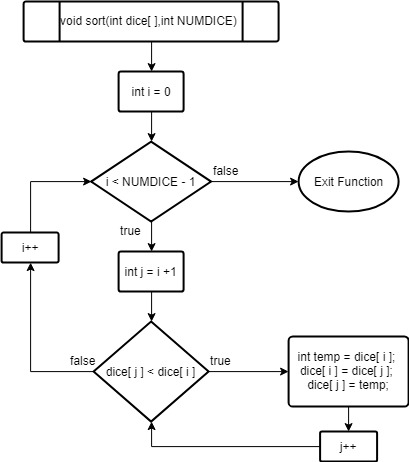
**FLOWCHART**

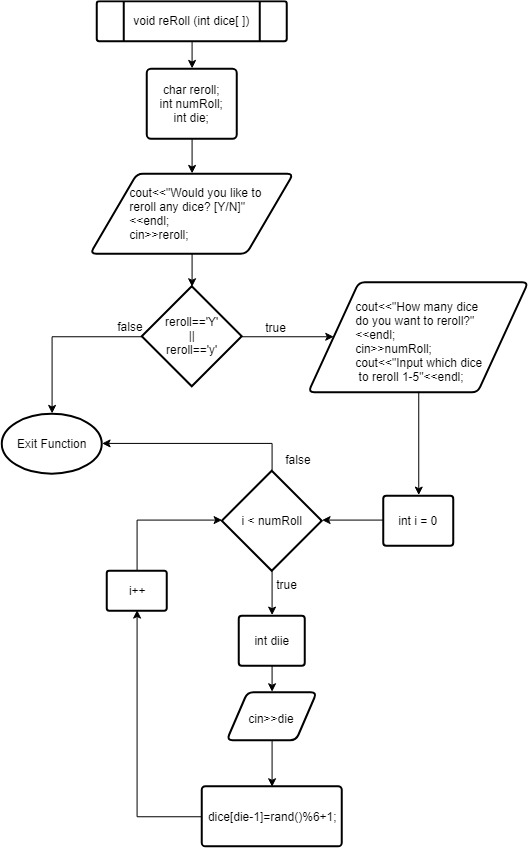
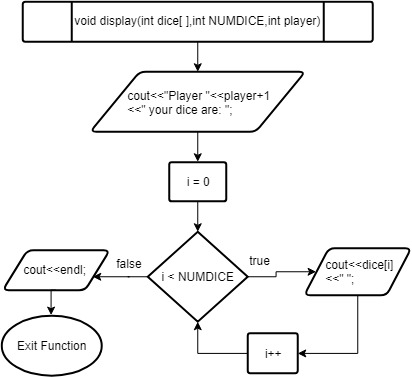
(Full size view available with attached file)

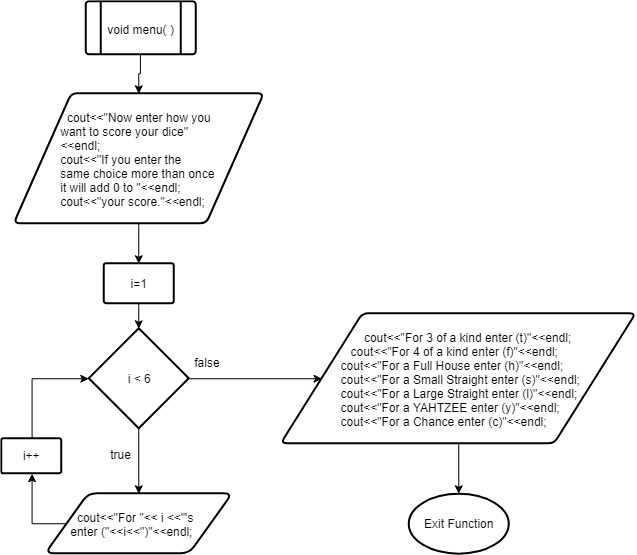
This picture constitutes *main*

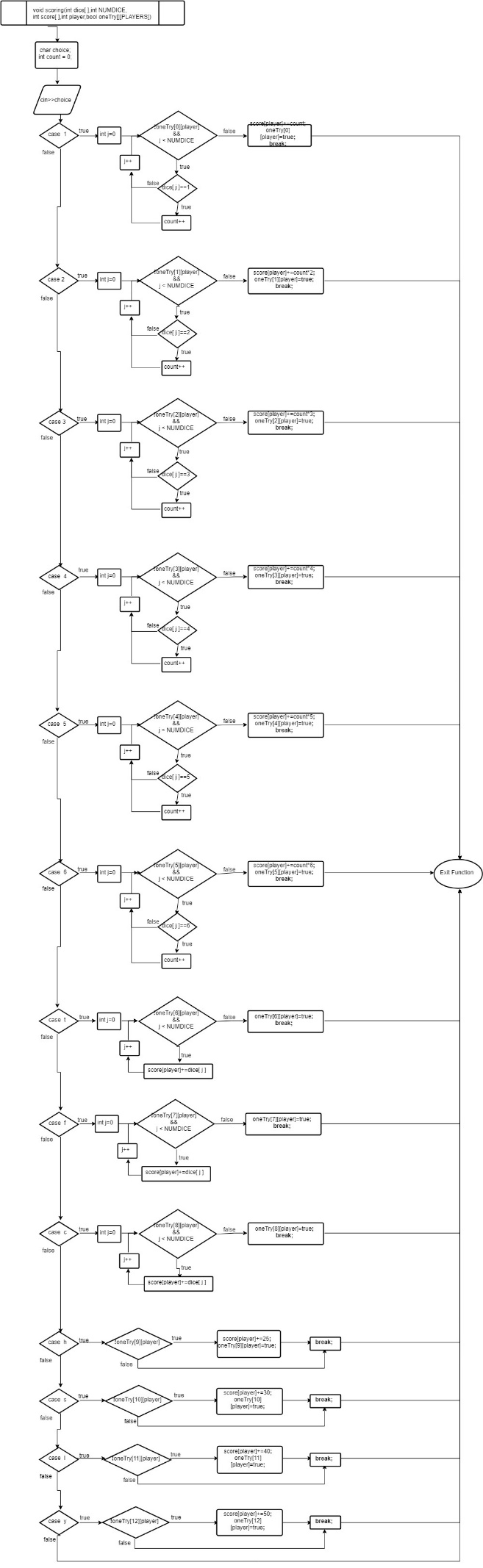
**** The rest are the functions included in *main*

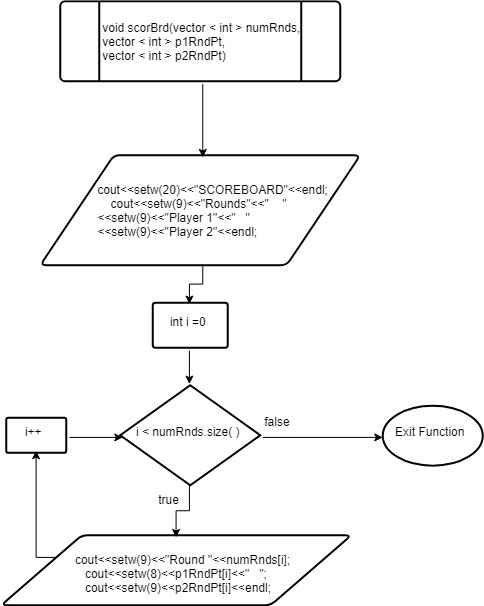
****

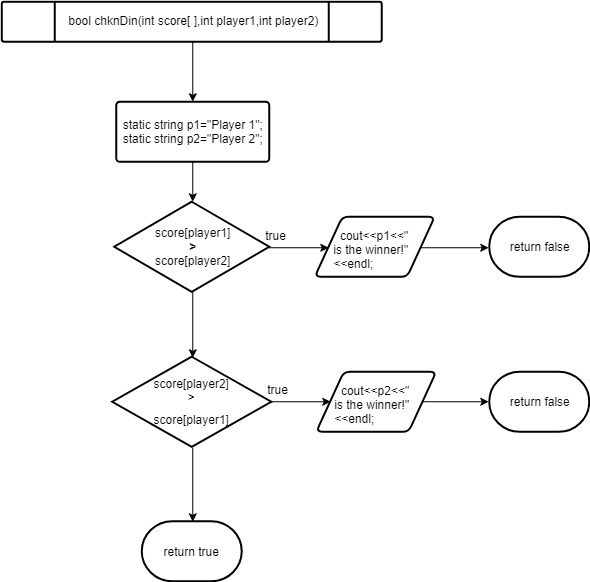


****

****

**(this is the scoring function. Full view available in attached jpg file)**

****

****

**CONSTRUCTS / CONCEPTS USED (CHECKLIST)**

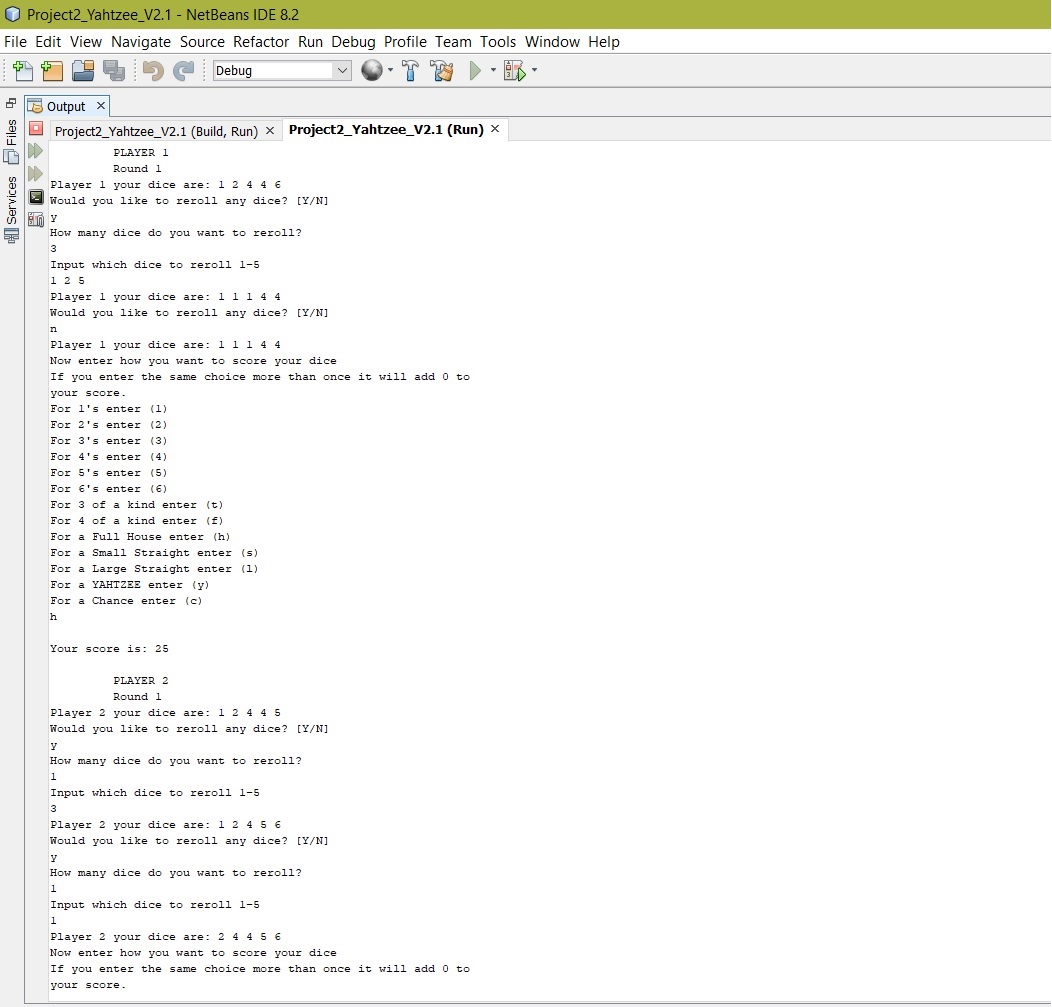
*PROJECT 1 CHECKLIST REVISIT STARTS ON NEXT PAGE*

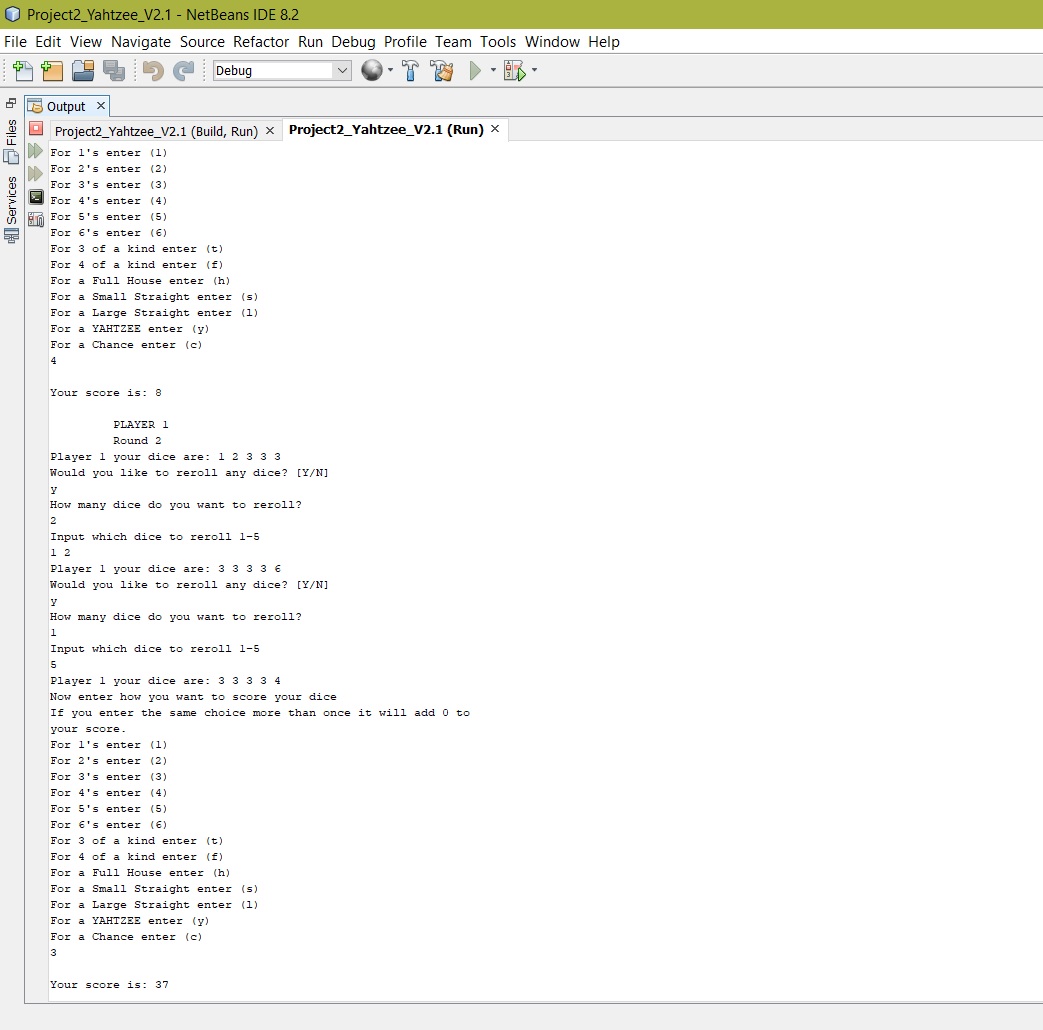
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chapter** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 2 | 2 | cout |  |  |  |
|  | 3 | libraries | 9 - 15 | 5 | iostream, iomanip, cmath,  cstdlib, fstream, string, ctime |
|  | 4 | variables/literals |  |  | No variables in global area, failed project! |
|  | 5 | Identifiers |  |  |  |
|  | 6 | Integers | 52 - 56 | 1 |  |
|  | 7 | Characters | 41, 45 | 1 |  |
|  | 8 | Strings | 263, 264 | 1 |  |
|  | 9 | Floats No Doubles | 66, 131 | 1 | Using doubles will fail the project, floats OK! |
|  | 10 | Bools | 61 | 1 |  |
|  | 11 | Sizeof \*\*\*\*\* |  |  |  |
|  | 12 | Variables 7 characters or less |  |  | All variables <= 7 characters |
|  | 13 | Scope \*\*\*\*\* No Global Variables |  |  |  |
|  | 14 | Arithmetic operators |  |  |  |
|  | 15 | Comments 20%+ |  | 2 | Model as pseudo code |
|  | 16 | Named Constants | 22, 50, 51 |  | All Local, only Conversions/Physics/Math in Global area |
|  | 17 | Programming Style \*\*\*\*\* Emulate |  |  | Emulate style in book/in class repositiory |
|  |  |  |  |  |  |
| 3 | 1 | cin | 122, 189, 192, 196 |  |  |
|  | 2 | Math Expression | 197, 225 |  |  |
|  | 3 | Mixing data types \*\*\*\* |  |  |  |
|  | 4 | Overflow/Underflow \*\*\*\* |  |  |  |
|  | 5 | Type Casting | 132 | 1 |  |
|  | 6 | Multiple assignment \*\*\*\*\* |  |  |  |
|  | 7 | Formatting output | 135, 279 - 285 | 1 |  |
|  | 8 | Strings | 290, 291 | 1 |  |
|  | 9 | Math Library | 132 | 1 | All libraries included have to be used |
|  | 10 | Hand tracing \*\*\*\*\*\* |  |  |  |
|  |  |  |  |  |  |
| 4 | 1 | Relational Operators |  |  |  |
|  | 2 | if | 120, 190 | 1 | Independent if |
|  | 4 | If-else | 296-399 | 1 |  |
|  | 5 | Nesting | 49-120, 190,194 | 1 |  |
|  | 6 | If-else-if | 292-296 | 1 |  |
|  | 7 | Flags \*\*\*\*\* |  |  |  |
|  | 8 | Logical operators | 123, 190, 223 | 1 |  |
|  | 11 | Validating user input |  | 1 |  |
|  | 13 | Conditional Operator |  | 1 |  |
|  | 14 | Switch | 221-274 | 1 |  |
|  |  |  |  |  |  |
| 5 | 1 | Increment/Decrement | 66,79 | 1 |  |
|  | 2 | While | 123 | 1 | Technicially… |
|  | 5 | Do-while | 49-123 | 1 |  |
|  | 6 | For loop | 66, 70, | 1 |  |
|  | 11 | Files input/output both | 45-48, 65 | 2 |  |
|  | 12 | No breaks in loops \*\*\*\*\*\* |  |  | Failed Project if included |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not required to show |  |  | Total | 30 |  |

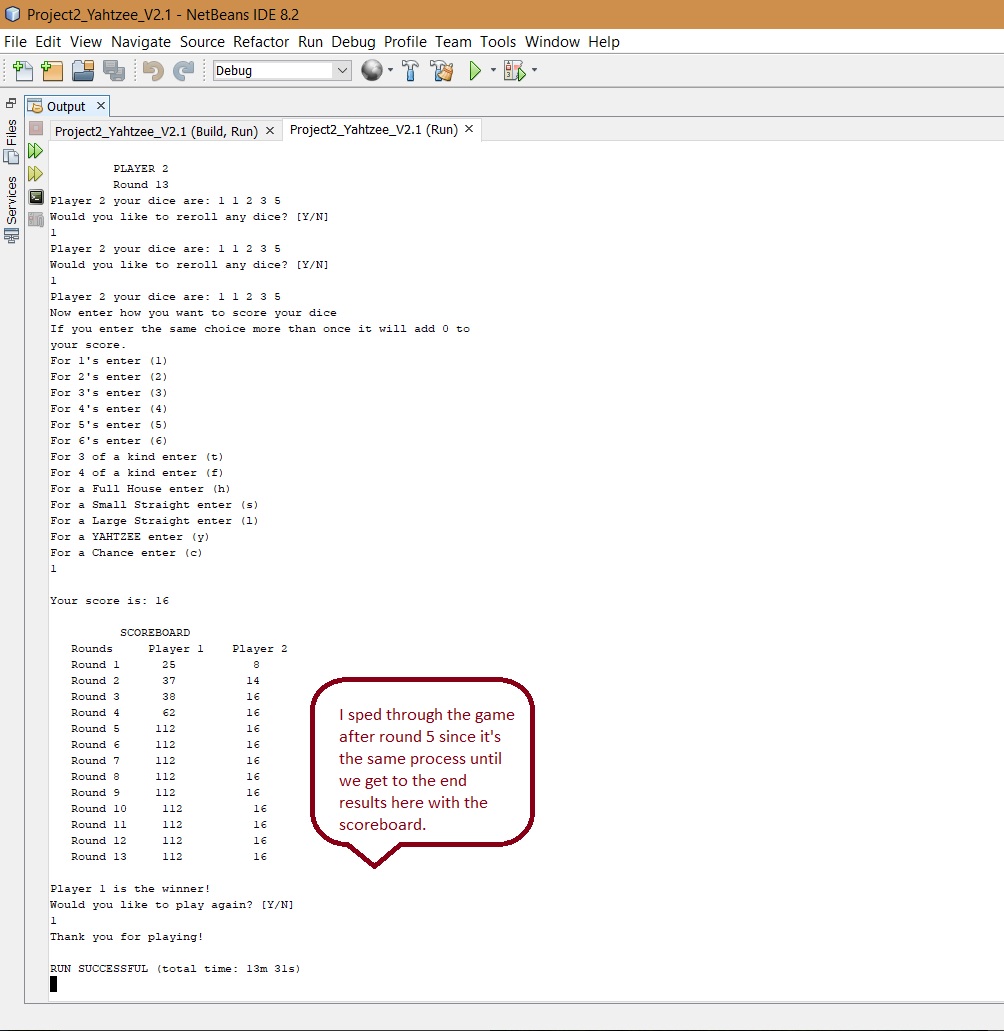
**PROJECT 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Chap** | **Section** | **Topic** | **Where Line #''s** | **Pts** | **Notes** |
| 6 |  | Functions |  |  |  |
|  | 3 | Function Prototypes | 25 - 33 | 4 | Always use prototypes |
|  | 5 | Pass by Value | 71 | 4 |  |
|  | 8 | return | 294, 298 | 4 | A value from a function |
|  | 9 | returning boolean | 300 | 4 |  |
|  | 10 | Global Variables |  | XXX | Do not use global variables -100 pts |
|  | 11 | static variables | 290, 291 | 4 |  |
|  | 12 | defaulted arguments | 26 | 4 |  |
|  | 13 | pass by reference | 71,74,98 | 4 | Anytime calling a function and passing  an array is pass by reference |
|  | 14 | overloading | 27, 28 | 5 |  |
|  | 15 | exit() function | 142 | 4 |  |
| 7 |  | Arrays |  |  |  |
|  | 1 to 6 | Single Dimensioned Arrays | 25-29, 33 | 3 |  |
|  | 7 | Parallel Arrays | 52, 53 | 2 |  |
|  | 8 | Single Dimensioned as Function Arguments | 71, 80 | 2 |  |
|  | 9 | 2 Dimensioned Arrays | 57 | 2 | Emulate style in book/in class repositiory |
|  | 12 | STL Vectors | 32 | 2 |  |
|  |  | Passing Arrays to and from Functions | 71 | 5 |  |
|  |  | Passing Vectors to and from Functions | 118 | 5 |  |
|  |  |  |  |  |  |
| 8 |  | Searching and Sorting Arrays |  |  |  |
|  | 3 | Bubble Sort | 27,74 | 4 |  |
|  | 3 | Selection Sort | 28,96 | 4 |  |
|  | 1 | Linear or Binary Search | 224 | 4 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| \*\*\*\*\*\* Not required to show |  |  | Total | 70 | Other 30 points from Proj 1 first sheet tab |

**SAMPLE OUTPUT**

****

****

****

**PROGRAM**

/\*

\* File: main.cpp

\* Author: Elise Chue

\* Created on May 31, 1:53 PM

\* Purpose: Yahtzee revamped for Project 2 using functions and such version 2

\*/

//System Libraries Here

#include <iostream>

#include <vector> //vectors

#include <cstdlib> //random function

#include <ctime> //rand num seed

#include <iomanip> //format library

#include <cmath> //math library for pow

#include <fstream> //I/O files

using namespace std;

//User Libraries Here

//Global Constants Only, No Global Variables

//Math, Physics, Science, Conversions, 2-D Array Columns

const int PLAYERS=2; //FOR 2D ARRAY COLUMN SIZE I SWEAR

//Function Prototypes

void roll(int [],int); //input 1 position and roll that die

void display(int [],int,int=0); //displays the 5 dice

void sort(int [],int); //sorts dice via markSort which is very similar to bubble sort

void sort(int [],int,char); //a select sort function to showcase overload concept

void reRoll(int []); //inputs which dice are going to be rerolled

void menu(); //displays categories for scoring

void scoring(int [],int,int [],int,bool [][PLAYERS]); //does actual scoring

void scorBrd(vector < int >,vector < int >,vector < int >); //prints out score each round

bool chknDin(int [],int,int);//winner winner chicken dinner

//Program Execution Begins Here

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

//set random number seed

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

//Declare all Variables, no doubles

char replay; //choose to replay the game

ifstream in; //Input File stream

ofstream out; //output file

string fileIn; //file name

char fileOut[]="Trivia.dat"; //uses cmath, calculates, and shows in out file

fileIn="Rounds.dat"; //will input the totRnd (total rounds that will be played in a game)

in.open(fileIn.c\_str());

out.open(fileOut);

do{

const int NUMDICE=5; //num of dice used for the entire game is 5

const int OPTIONS=13; //number of options to score your roll

int p1Dice[NUMDICE]; //array of 5 dice for player 1

int p2Dice[NUMDICE]; //array of 5 dice for player 2

int score[PLAYERS]={0,0};//player's score array position is 0 for player 1, 1 for player 2

int player1=0; //used for calling functions

int player2=1; //used for calling functions

bool oneTry [OPTIONS][PLAYERS] = { };

vector < int > numRnds; //vector to count the number of rounds

vector < int > p1RndPt; //points at end of round for player 1

vector < int > p2RndPt; //points at end of round for player 2

bool tie=false;

int totRnd; //file is set for 13 rounds (2 for testing purposes)

//Input or initialize values

in>>totRnd;

for(float round=0;round<totRnd;round++){ //Increments the amount of Rounds played

//Roll the first set for player 1

cout<<" PLAYER 1 "<<endl;

cout<<" Round "<<round+1<<endl; //Displays which round youre on

for(int i=0;i<NUMDICE;i++){ //rolls 5 dice

roll(p1Dice,i);

}

//Sort p1Dice

sort(p1Dice,NUMDICE); //sorts the dice smallest to highest

//display p1 Dice //so its easier to see what you have going for you

display(p1Dice,NUMDICE);

//reroll p1 Dice

for(int i=1;i<=2;i++){ //two turns to reroll p1Dice

reRoll(p1Dice); //reroll

sort(p1Dice,NUMDICE); //sort

display(p1Dice,NUMDICE,player1);//display

}

//menu and scoring for p1

menu(); //calls the score category menu function

scoring(p1Dice,NUMDICE,score,player1,oneTry);

cout<<endl<<"Your score is: "<<score[player1]<<endl<<endl;

cout<<" PLAYER 2 "<<endl;

cout<<" Round "<<round+1<<" "<<endl;

//Roll the first set for player 2

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

roll(p2Dice,i);

}

//Sort p2Dice

sort(p2Dice,NUMDICE,'s'); //s for "selection sort" as proof of concept for overloading

//display p2 Dice

display(p2Dice,NUMDICE,player2);

//reroll p2 Dice

for(int i=1;i<=2;i++){ //two turns to reroll p2Dice

reRoll(p2Dice); //reroll

sort(p2Dice,NUMDICE); //selection sort

display(p2Dice,NUMDICE,player2);//display

}

//menu and scoring for p2 which is essentially the same as p1

menu();

scoring(p2Dice,NUMDICE,score,player2,oneTry);

cout<<endl<<"Your score is: "<<score[player2]<<endl<<endl;

numRnds.push\_back(round+1);

//track round score for p1 and p2

p1RndPt.push\_back(score[player1]);

p2RndPt.push\_back(score[player2]);

}

//Display Scoreboard

scorBrd(numRnds,p1RndPt,p2RndPt);

//Display Winner

if(chknDin(score,player1,player2))cout<<"It is a TIE!"<<endl;

cout<<"Would you like to play again? [Y/N]"<<endl;

cin>>replay;

}while (replay=='y'||replay=='Y'); //end of do while loop

cout<<"Thank you for playing!"<<endl;

//out file Trivia.dat, will show calculations in that file

out<<"Let's calculate the odds of getting a Yahtzee in 1 roll!"<<endl;

out<<"Each dice has a 1/6 chance of turning up #1-6"<<endl;

out<<"Given the first dice is the number you want to match:"<<endl;

float prob; //probability of yahtzee in one roll (decimal)

prob=pow((static\_cast<float>(1)/6),4);

out<<"That comes out to (1/6)^4 (the expo. being the remaining 4 dice)"

" = "<<prob<<endl;

out<<"which equates to "<<setprecision(1)<<prob\*100<<"% chance"<<endl;

//convert to a percentage which is less than 1 percent!

//Close the file

in.close();

out.close();

exit(0); //this goes right to return 0 which ends the function. i didnt know where else to put it

//Exit Program!

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// FUNCTIONS START HERE

void roll(int dice[],int die){

dice[die]=rand()%6+1;

}

void sort(int dice[],int NUMDICE){ //Bubble/Mark sort swaps lowest number to first position

for(int i=0;i<NUMDICE-1;i++){ //then repeat until all numbers are in order

for(int j=i+1;j<NUMDICE;j++){

if(dice[j]<dice[i]){ //swap logic

int temp=dice[i];

dice[i]=dice[j];

dice[j]=temp;

}

}

}

}

void sort(int dice[],int NUMDICE,char s){ //Selection sort

for(int lstSize=0;lstSize<NUMDICE-1;lstSize++){ //ListSize

char small=dice[lstSize];

char indxSml=lstSize;

for(int findSml=lstSize+1;findSml<NUMDICE;findSml++){

if(small>dice[findSml]){

small=dice[findSml];

indxSml=findSml;

}

}

char temp=dice[lstSize]; //swap logic

dice[lstSize]=dice[indxSml];

dice[indxSml]=temp;

}

}

void display(int dice[],int NUMDICE,int player){

cout<<"Player "<<player+1<<" your dice are: ";

for(int i=0;i<NUMDICE;i++){ //display all 5 dice

cout<<dice[i]<<" ";

}

cout<<endl;

}

void reRoll(int dice[]){

cout<<"Would you like to reroll any dice? [Y/N]"<<endl;

char reroll; //decides whether to reroll or not

int numRoll; //number of dice to be rerolled

cin>>reroll;

if(reroll=='Y'||reroll=='y'){ //if they want to reroll

cout<<"How many dice do you want to reroll?"<<endl;

cin>>numRoll; //how MANY dice are they rerolling

cout<<"Input which dice to reroll 1-5"<<endl;

for(int i=0;i<numRoll;i++){ //loop numRoll times

int die; //WHICH die they want to reroll

cin>>die;

dice[die-1]=rand()%6+1;//reroll the dice

}

}

}

void menu(){

cout<<"Now enter how you want to score your dice"<<endl;

cout<<"If you enter the same choice more than once it will add 0 to "<<endl;

cout<<"your score."<<endl;

for(int i=1;i<=6;i++){

cout<<"For "<<i<<"'s enter ("<<i<<")"<<endl; //increments for value 1 - 6 options

}

cout<<"For 3 of a kind enter (t)"<<endl;

cout<<"For 4 of a kind enter (f)"<<endl;

cout<<"For a Full House enter (h)"<<endl;

cout<<"For a Small Straight enter (s)"<<endl;

cout<<"For a Large Straight enter (l)"<<endl;

cout<<"For a YAHTZEE enter (y)"<<endl;

cout<<"For a Chance enter (c)"<<endl;

}

void scoring(int dice[],int NUMDICE,int score[],int player, //calculates score

bool oneTry[][PLAYERS]){ //validates whether or not we've filled a score category already

char choice;

int count=0;

cin>>choice;

switch(choice){

case '1':{ //UPPER SECTION score categories

for(int j=0;!oneTry[0][player]&&j<NUMDICE;j++){

if(dice[j]==1)count++; //searches the array for a specific #, in this case: 1

}score[player]+=count;oneTry[0][player]=true;break; //Cumulative score

}

case '2':{ //ditto with case 1 but w/ 2's

for(int j=0;!oneTry[1][player]&&j<NUMDICE;j++){

if(dice[j]==2)count++;

}score[player]+=(count\*2);oneTry[1][player]=true;break; //multiplies how many 2's by 2

}

case '3':{ //etc.

for(int j=0;!oneTry[2][player]&&j<NUMDICE;j++){

if(dice[j]==3)count++;

}score[player]+=(count\*3);oneTry[2][player]=true;break;

}

case '4':{

for(int j=0;!oneTry[3][player]&&j<NUMDICE;j++){

if(dice[j]==4)count++;

}score[player]+=(count\*4);oneTry[3][player]=true;break;

}

case '5':{

for(int j=0;!oneTry[4][player]&&j<NUMDICE;j++){

if(dice[j]==5)count++;

}score[player]+=(count\*5);oneTry[4][player]=true;break;

}

case '6':{

for(int j=0;!oneTry[5][player]&&j<NUMDICE;j++){

if(dice[j]==6)count++;

}score[player]+=(count\*6);oneTry[5][player]=true;break;

}

case 't':{ //start of "Lower Section" score categories

for(int j=0;!oneTry[6][player]&&j<NUMDICE;j++){

score[player]+=dice[j]; //Three of a kind Add all dice

}oneTry[6][player]=true;break;

}

case 'f':{

for(int j=0;!oneTry[7][player]&&j<NUMDICE;j++){

score[player]+=dice[j]; //Four of a kind Add all dice

}oneTry[7][player]=true;break;

}

case 'c':{

for(int j=0;!oneTry[8][player]&&j<NUMDICE;j++){

score[player]+=dice[j]; //Chance Add all dice

}oneTry[8][player]=true;break;

}

case 'h':{if(!oneTry[9][player]){

score[player]+=25;oneTry[9][player]=true;}break;} //Full House Score 25

case 's':{if(!oneTry[10][player]){

score[player]+=30;oneTry[10][player]=true;}break;} //Small Straight Score 30

case 'l':{if(!oneTry[11][player]){

score[player]+=40;oneTry[11][player]=true;}break;} //Large Straight Score 40

case 'y':{if(!oneTry[12][player]){

score[player]+=50;oneTry[12][player]=true;}break;} //Yahtzee Score 50

}

}

void scorBrd(vector < int > numRnds,vector < int > p1RndPt, //Displays the Scores

vector < int > p2RndPt){

cout<<setw(20)<<"SCOREBOARD"<<endl;

cout<<setw(9)<<"Rounds"<<" "<<setw(9)<<"Player 1"<<" "//displays each round's cumulative score

<<setw(9)<<"Player 2"<<endl;

for(int i=0;i<numRnds.size();i++){ //"rounds" increases with each turn hence the .size

cout<<setw(9)<<"Round "<<numRnds[i];

cout<<setw(8)<<p1RndPt[i]<<" ";

cout<<setw(9)<<p2RndPt[i]<<endl;

}

}

bool chknDin(int score[],int player1,int player2){ //determines winner

cout<<endl;

static string p1="Player 1"; //needed a static and a string

static string p2="Player 2";

if(score[player1]>score[player2]){ //simple score comparison

cout<<p1<<" is the winner!"<<endl;

return false; //needed a bool return

}

else if(score[player2]>score[player1]){

cout<<p2<<" is the winner!"<<endl;

return false;

}

else return true;

}

**REFERENCES**

Dr. Lehr’s Lectures and Code Repository

Gaddis 9th Ed. Textbook

Savitch 9th Ed. Textbook