Yahtzee Scoring Module

Requirements:

The program should accept as arguments an integer array of 5 dice rolls between 1 and 6 (inclusive), as well as a string containing the category to be scored. If a category has already been scored, the program should output an error message and leave the score for that category unchanged.

For the categories ‘Aces’, ‘Twos’, ‘Threes’, ‘Fours’, ‘Fives’, and ‘Sixes’, the program should count the number of those rolls in the integer array and assign that count multiplied by the value of the roll to the score for that category. Once these categories have been scored, the program should assign the total score for these six categories to total score (upper), and assign 35 to the bonus if that total is greater than or equal to 63, or 0 to the bonus if the total is less than 63. It should then add the bonus score and total score (upper) together and assign the value to the total of upper section score.

For the ‘3 of a Kind’ and ‘4 of a Kind’ categories, the program should count each number in the integer array, and if any of these counts is greater than or equal to 3 or 4 respectively, assign the total of the 5 roll values to the respective category score.

For the ‘Full House’ category, the program should count each number in the integer array, and assign a score of 25 to the category if there is a count of 2 and a count of 3.

For the ‘Small Straight’ and ‘Large Straight’ categories, the program should check the array to see if it contains 4 or 5 adjacent values, respectively, and in any order, and assign a score of 30 or 40 respectively.

For the ‘Yahtzee’ category, the program should count each value in the array. If there is a count of 5, the program should assign a score of 50 to the category.

For the ‘Chance’ category, the program should total the values of the 5 rolls and assign that total to the score for the category.

For the ‘Yahtzee Bonus’ category, the program should check each time a roll is submitted whether there is a 50 scored in the ‘Yahtzee’ category. If there is, the program should check for 5 identical roll values, and if 5 identical values exist, add 100 to the ‘Yahtzee Bonus’ category.

Once the aforementioned categories are scored, or if every aforementioned category besides ‘Yahtzee Bonus’ is scored, the program should assign a 0 to the ‘Yahtzee Bonus’ score if it is unscored, and total the scores for ‘3 of a Kind’, ‘4 of a Kind’, ‘Full House’, ‘Small Straight’, ‘Large Straight’, ‘Yahtzee’, ‘Chance’, and ‘Yahtzee Bonus’, and assign the total to the total of lower section score. It should then add the total of upper section score to the total of lower section score, and assign this value to the grand total score.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testing Plan (by Game) | | | | | |
| # | Test Details | Inputs (in Test Cases.txt) | Expected Output | Actual Output | Pass/Fail |
| Game 1, Testing a Minimum Score Game | | | | | |
| 1 | Test zero 1’s | Aces 23565000 | 0 | 0 | Pass |
| 2 | Test zero 2’s | Twos 13146000 | 0 | 0 | Pass |
| 3 | Test zero 3’s | Threes 24251000 | 0 | 0 | Pass |
| 4 | Test zero 4’s | Fours 35362000 | 0 | 0 | Pass |
| 5 | Test zero 5’s | Fives 46413000 | 0 | 0 | Pass |
| 6 | Test zero 6’s | Sixes 51524000 | 0 | 0 | Pass |
| 7 | Testing with two of a kind | ThreeOfKind 66112000 | 0 | 0 | Pass |
| 8 | Testing with three of a kind | FourOfKind 11166000 | 0 | 0 | Pass |
| 9 | Testing with three of a kind (no pair) | FullHouse 55561000 | 0 | 0 | Pass |
| 10 | Testing with sequence of 3 | SmallStraight 12356000 | 0 | 0 | Pass |
| 11 | Testing with sequence of 4 | LargeStraight 34561000 | 0 | 0 | Pass |
| 12 | Testing with full house | Yahtzee 66655000 | 0 | 0 | Pass |
| 13 | Testing with Yahtzee | Chance 11111005 | 5 | 5 | Pass |
| 14 | Test Total Score(Upper) | TotalScore  00001000 | 0 | 0 | Pass |
| 15 | Test Bonus(Upper) | BonusUpper  00001000 | 0 | 0 | Pass |
| 16 | Test Total of Upper Section | TotalUpper  00001000 | 0 | 0 | Pass |
| 17 | Test Yahtzee Bonus (Yahtzee with 0 score in Yahtzee category) | YahtzeeBonus  00001000 | 0 | 0 | Pass |
| 18 | Test Total of Lower Section | TotalLower  00001005 | 5 | 5 | Pass |
| 19 | Test Grand Total | GrandTotal  00001005 | 5 | 5 | Pass |
| Game 2, Testing with Max/High Scores | | | | | |
| 20 | Test five 1’s | NewGame Aces 11111005 | 5 | 5 | Pass |
| 21 | Test five 2’s | Twos 22222010 | 10 | 10 | Pass |
| 22 | Test five 3’s | Threes 33333015 | 15 | 15 | Pass |
| 23 | Test five 4’s | Fours  44444020 | 20 | 20 | Pass |
| 24 | Test five 5’s | Fives  55555025 | 25 | 25 | Pass |
| 25 | Test five 6’s | Sixes  66666030 | 30 | 30 | Pass |
| 26 | Testing with five of a kind | ThreeOfKind  66666030 | 30 | 30 | Pass |
| 27 | Testing with five of a kind | FourOfKind  66666030 | 30 | 30 | Pass |
| 28 | Testing with pair and three of a kind | FullHouse  66555025 | 25 | 25 | Pass |
| 29 | Testing with sequence of 5 | SmallStraight  23456030 | 30 | 30 | Pass |
| 30 | Testing with sequence of 5 | LargeStraight  12345040 | 40 | 40 | Pass |
| 31 | Testing with Yahtzee of 1’s | Yahtzee  11111050 | 50 | 50 | Pass |
| 32 | Testing with Yahtzee | Chance  66666030 | 30 | 30 | Pass |
| 33 | Test Total Score(Upper) | TotalScore  00001105 | 105 | 105 | Pass |
| 34 | Test Bonus(Upper) | BonusUpper  00001035 | 35 | 35 | Pass |
| 35 | Test Total of Upper Section | TotalUpper  00001140 | 140 | 140 | Pass |
| 36 | Test Yahtzee Bonus (Yahtzee with 50 score in Yahtzee category) | YahtzeeBonus  00001100 | 100 | 100 | Pass |
| 37 | Test Total of Lower Section | TotalLower  00001335 | 335 | 335 | Pass |
| 38 | Test Grand Total | GrandTotal  00001475 | 475 | 475 | Pass |
| Game 3, Alternate Test Cases | | | | | |
| 39 | Test one 1 | NewGame Aces  66661001 | 1 | 1 | Pass |
| 40 | Test one 2 | Twos  25555002 | 2 | 2 | Pass |
| 41 | Test one 3 | Threes  61345003 | 3 | 3 | Pass |
| 42 | Test one 4 | Fours  12345004 | 4 | 4 | Pass |
| 43 | Test one 5 | Fives  25222005 | 5 | 5 | Pass |
| 44 | Test one 6 | Sixes  65432006 | 6 | 6 | Pass |
| 45 | Testing with three of a kind | ThreeOfKind  11166015 | 15 | 15 | Pass |
| 46 | Testing with four of a kind | FourOfKind  44443019 | 19 | 19 | Pass |
| 47 | Testing with pair (no three of a kind) | FullHouse  66123000 | 0 | 0 | Pass |
| 48 | Testing with a sequence of 4 | SmallStraight  14563030 | 30 | 30 | Pass |
| 49 | Testing with sequence out of order | LargeStraight  24315040 | 40 | 40 | Pass |
| 50 | Testing with Yahtzee of 6’s | Yahtzee  66666050 | 50 | 50 | Pass |
| 51 | Testing with large straight | Chance  23456020 | 20 | 20 | Pass |
| 52 | Test Total Score(Upper) | TotalScore  00001021 | 21 | 21 | Pass |
| 53 | Test Bonus(Upper) | BonusUpper  00001000 | 0 | 0 | Pass |
| 54 | Test Total of Upper Section | TotalUpper  00001021 | 21 | 21 | Pass |
| 55 | Test Yahtzee Bonus | YahtzeeBonus  00001000 | 0 | 0 | Pass |
| 56 | Test Total of Lower Section | TotalLower  00001174 | 174 | 174 | Pass |
| 57 | Test Grand Total | GrandTotal  00001195 | 195 | 195 | Pass |
| Game 4, Additional Cases | | | | | |
| 58 | Test that score cannot be overwritten | NewGame Aces  23565!!!  Aces  11116!!! | “ERROR: Category has already been scored.” | “ERROR: Category has already been scored.” | Pass |
| 59 | Test Full House with 5 of Kind | FullHouse  66666000 | 0 | 0 | Pass |
| Testing Plan (by Scoring Category) | | | | | |
| # | Test Details | Inputs | Expected Output | Actual Output | Pass/Fail |
| Testing Aces | | | | | |
| 1 | Test zero 1’s | 2-3-5-6-5, Aces | 0 | 0 | Pass |
| 20 | Test five 1’s | 1-1-1-1-1, Aces | 5 | 5 | Pass |
| 39 | Test one 1 | 6-6-6-6-1, Aces | 1 | 1 | Pass |
| 58 | Test that score cannot be overwritten | 2-3-5-6-5, Aces  1-1-1-1-6, Aces | “ERROR: Category has already been scored.” | “ERROR: Category has already been scored.” | Pass |
| Testing Twos | | | | | |
| 2 | Test zero 2’s | 1-3-1-4-6, Twos | 0 | 0 | Pass |
| 21 | Test five 2’s | 2-2-2-2-2, Twos | 10 | 10 | Pass |
| 40 | Test one 2 | 2-5-5-5-5, Twos | 2 | 2 | Pass |
| Testing Threes | | | | | |
| 3 | Test zero 3’s | 2-4-2-5-1, Threes | 0 | 0 | Pass |
| 22 | Test five 3’s | 3-3-3-3-3, Threes | 15 | 15 | Pass |
| 41 | Test one 3 | 6-1-3-4-5, Threes | 3 | 3 | Pass |
| Testing Fours | | | | | |
| 4 | Test zero 4’s | 3-5-3-6-2, Fours | 0 | 0 | Pass |
| 23 | Test five 4’s | 4-4-4-4-4, Fours | 20 | 20 | Pass |
| 42 | Test one 4 | 1-2-3-4-5, Fours | 4 | 4 | Pass |
| Testing Fives | | | | | |
| 5 | Test zero 5’s | 4-6-4-1-3, Fives | 0 | 0 | Pass |
| 24 | Test five 5’s | 5-5-5-5-5, Fives | 25 | 25 | Pass |
| 43 | Test one 5 | 2-5-2-2-2, Fives | 5 | 5 | Pass |
| Testing Sixes | | | | | |
| 6 | Test zero 6’s | 5-1-5-2-4, Sixes | 0 | 0 | Pass |
| 25 | Test five 6’s | 6-6-6-6-6, Sixes | 30 | 30 | Pass |
| 44 | Test one 6 | 6-5-4-3-2, Sixes | 6 | 6 | Pass |
| Testing Total Score (Upper) | | | | | |
| 14 | Test Total Score(Upper) | TotalScore | 0 | 0 | Pass |
| 33 | Test Total Score(Upper) | TotalScore | 105 | 105 | Pass |
| 52 | Test Total Score(Upper) | TotalScore | 21 | 21 | Pass |
| Testing Bonus (Upper) | | | | | |
| 15 | Test Bonus(Upper) | BonusUpper | 0 | 0 | Pass |
| 34 | Test Bonus(Upper) | BonusUpper | 35 | 35 | Pass |
| 53 | Test Bonus(Upper) | BonusUpper | 0 | 0 | Pass |
| Testing Total of Upper Section | | | | | |
| 16 | Test Total of Upper Section | TotalUpper | 0 | 0 | Pass |
| 35 | Test Total of Upper Section | TotalUpper | 140 | 140 | Pass |
| 54 | Test Total of Upper Section | TotalUpper | 21 | 21 | Pass |
| Testing Three of a Kind | | | | | |
| 7 | Testing with two of a kind | 6-6-1-1-2, ThreeOfKind | 0 | 0 | Pass |
| 26 | Testing with five of a kind | 6-6-6-6-6, ThreeOfKind | 30 | 30 | Pass |
| 45 | Testing with three of a kind | 1-1-1-6-6, ThreeOfKind | 15 | 15 | Pass |
| Testing Four of a Kind | | | | | |
| 8 | Testing with three of a kind | 1-1-1-6-6, FourOfKind | 0 | 0 | Pass |
| 27 | Testing with five of a kind | 6-6-6-6-6, FourOfKind | 30 | 30 | Pass |
| 46 | Testing with four of a kind | 4-4-4-4-3, FourOfKind | 19 | 19 | Pass |
| Testing Full House | | | | | |
| 9 | Testing with three of a kind (no pair) | 5-5-5-6-1, FullHouse | 0 | 0 | Pass |
| 28 | Testing with pair and three of a kind | 6-6-5-5-5, FullHouse | 25 | 25 | Pass |
| 47 | Testing with pair (no three of a kind) | 6-6-1-2-3, FullHouse | 0 | 0 | Pass |
| 59 | Test Full House with 5 of Kind | 6-6-6-6-6, Full House | 0 | 0 | Pass |
| Testing Small Straight | | | | | |
| 10 | Testing with sequence of 3 | 1-2-3-5-6, SmallStraight | 0 | 0 | Pass |
| 29 | Testing with sequence of 5 | 2-3-4-5-6, SmallStraight | 30 | 30 | Pass |
| 48 | Testing with a sequence of 4 | 1-4-5-6-3, SmallStraight | 30 | 30 | Pass |
| Testing Large Straight | | | | | |
| 11 | Testing with sequence of 4 | 3-4-5-6-1, LargeStraight | 0 | 0 | Pass |
| 30 | Testing with sequence of 5 | 1-2-3-4-5, LargeStraight | 40 | 40 | Pass |
| 49 | Testing with sequence out of order | 2-4-3-1-5, LargeStraight | 40 | 40 | Pass |
| Testing Yahtzee | | | | | |
| 12 | Testing with full house | 6-6-6-5-5, Yahtzee | 0 | 0 | Pass |
| 31 | Testing with Yahtzee of 1’s | 1-1-1-1-1, Yahtzee | 50 | 50 | Pass |
| 50 | Testing with Yahtzee of 6’s | 6-6-6-6-6, Yahtzee | 50 | 50 | Pass |
| Testing Chance | | | | | |
| 13 | Testing with Yahtzee | 1-1-1-1-1, Chance | 5 | 5 | Pass |
| 32 | Testing with Yahtzee | 6-6-6-6-6, Chance | 30 | 30 | Pass |
| 51 | Testing with large straight | 2-3-4-5-6, Chance | 20 | 20 | Pass |
| Testing Yahtzee Bonus | | | | | |
| 17 | Test Yahtzee Bonus (Yahtzee with 0 score in Yahtzee category) | YahtzeeBonus | 0 | 0 | Pass |
| 36 | Test Yahtzee Bonus (Yahtzee with 50 score in Yahtzee category) | YahtzeeBonus | 100 | 100 | Pass |
| 55 | Test Yahtzee Bonus | YahtzeeBonus | 0 | 0 | Pass |
| Testing Total of Lower Section | | | | | |
| 18 | Test Total of Lower Section | TotalLower | 5 | 5 | Pass |
| 37 | Test Total of Lower Section | TotalLower | 335 | 335 | Pass |
| 56 | Test Total of Lower Section | TotalLower | 174 | 174 | Pass |
| Testing Grand Total | | | | | |
| 19 | Test Grand Total | GrandTotal | 5 | 5 | Pass |
| 38 | Test Grand Total | GrandTotal | 475 | 475 | Pass |
| 57 | Test Grand Total | GrandTotal | 220 | 220 | Pass |

Testing Harness Output:

Test Case 1 Passed

Test Case 2 Passed

Test Case 3 Passed

Test Case 4 Passed

Test Case 5 Passed

Test Case 6 Passed

Test Case 7 Passed

Test Case 8 Passed

Test Case 9 Passed

Test Case 10 Passed

Test Case 11 Passed

Test Case 12 Passed

Test Case 13 Passed

Test Case 14 Passed

Test Case 15 Passed

Test Case 16 Passed

Test Case 17 Passed

Test Case 18 Passed

Test Case 19 Passed

Test Case 20 Passed

Test Case 21 Passed

Test Case 22 Passed

Test Case 23 Passed

Test Case 24 Passed

Test Case 25 Passed

Test Case 26 Passed

Test Case 27 Passed

Test Case 28 Passed

Test Case 29 Passed

Test Case 30 Passed

Test Case 31 Passed

Test Case 32 Passed

Test Case 33 Passed

Test Case 34 Passed

Test Case 35 Passed

Test Case 36 Passed

Test Case 37 Passed

Test Case 38 Passed

Test Case 39 Passed

Test Case 40 Passed

Test Case 41 Passed

Test Case 42 Passed

Test Case 43 Passed

Test Case 44 Passed

Test Case 45 Passed

Test Case 46 Passed

Test Case 47 Passed

Test Case 48 Passed

Test Case 49 Passed

Test Case 50 Passed

Test Case 51 Passed

Test Case 52 Passed

Test Case 53 Passed

Test Case 54 Passed

Test Case 55 Passed

Test Case 56 Passed

Test Case 57 Passed

Test Case 58 Passed

Test Case 59 Passed

Press any key to continue . . .

Assumptions and Procedure:

Most of the scoring categories were straightforward, except for the Full House category. The official rules state that a full house is “three of one number and two of another”, which implies that the pair and three of a kind cannot be the same number, so I have coded the program to score a yahtzee as 0 in the full house category. Since this program isn’t responsible for the die rolls, I am assuming that any dice rolling module will be tested so that it only rolls whole numbers 1-6, thus I have not checked the input for rolls for invalid numbers.

The testing harness reads input from the file in the following manner: The first line is read which contains the category which will be passed to the main program. The second line contains numbers representing the rolls and a 3 digit expected result. The first five numbers are the rolls, and the next 3 numbers are the expected result. Leading zeros should be used for the expected result; for example, if the expected result is 5, 005 should be entered into the testing harness’ input file. This input file is named Test Cases.txt. There should be no spaces in the text file.

When coding the testing harness, it was discovered that it was impossible to check the values for total score, bonus, and other non-category values. The program was modified with additional category inputs which return these values. When testing these values, any numerical value for rolls can be entered, as long as the 5 die rolls are not the same digit. This will cause 100 to be added erroneously to the yahtzee bonus if ‘50’ has been scored in the yahtzee category.

It was also discovered that because the program returns a value of -1 when an attempt is made to score a category twice, and the testing harness will not read a -1 since it is two separate characters, that an alternative method was needed to test whether the program doesn’t allow a category to be scored twice. This was solved by entering !!! as the expected value. These characters are converted to a negative integer, in which case a separate ‘if’ statement is triggered which compares the returned value to -1.

After each game, the line ‘NewGame’ should be added to Test Cases.txt. This causes the testing harness to create a new object of the scoring module where all categories are unscored.

//YahtzeeTestingHarness.cpp

#include "YahtzeeScoreModule.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

string testCaseFilename = "Test Cases.txt";

ifstream testCases;

testCases.open(testCaseFilename);

YahtzeeScoreModule \*scoreModule = new YahtzeeScoreModule();

int caseNumber = 1;

int rolls[5] = { 0,0,0,0,0 };

if (testCases.is\_open()) {

while (!testCases.eof()) {

string category;

getline(testCases, category);

if (category == "NewGame") {

delete scoreModule;

YahtzeeScoreModule \*scoreModule = new YahtzeeScoreModule();

getline(testCases, category);

}

string rollData;

getline(testCases, rollData);

for (int i = 0; i < 5; i++) {

rolls[i] = rollData[i] - '0';

}

int expectedResult = 0;

expectedResult = (((int)rollData[5] - '0') \* 100) + (((int)rollData[6]-'0') \* 10) + ((int)rollData[7] - '0');

if (expectedResult < 0) {

scoreModule->functionCaller(rolls, category);

getline(testCases, category);

getline(testCases, rollData);

for (int i = 0; i < 5; i++) {

rolls[i] = rollData[i] - '0';

}

int result = (scoreModule->functionCaller(rolls, category));

if (result == -1) {

cout << "Test Case " << caseNumber << " Passed\n";

}

else {

cout << "Test Case " << caseNumber << " Failed\n";

}

caseNumber++;

}

else {

int result = (scoreModule->functionCaller(rolls, category));

if (expectedResult == result) {

cout << "Test Case " << caseNumber << " Passed\n";

caseNumber++;

}

else if (expectedResult >= 0) {

cout << "Test Case " << caseNumber << " Failed: Expected " << expectedResult << ", got " << result << ".\n";

caseNumber++;

}

}

}

} else {

cout << "Error opening file.\n";

}

system("pause");

return 0;

}

Test Cases.txt Contents:

Aces

23565000

Twos

13146000

Threes

24251000

Fours

35362000

Fives

46413000

Sixes

51524000

ThreeOfKind

66112000

FourOfKind

11166000

FullHouse

55561000

SmallStraight

12356000

LargeStraight

34561000

Yahtzee

66655000

Chance

11111005

TotalScore

00001000

BonusUpper

00001000

TotalUpper

00001000

YahtzeeBonus

00001000

TotalLower

00001005

GrandTotal

00001005

NewGame

Aces

11111005

Twos

22222010

Threes

33333015

Fours

44444020

Fives

55555025

Sixes

66666030

ThreeOfKind

66666030

FourOfKind

66666030

FullHouse

66555025

SmallStraight

23456030

LargeStraight

12345040

Yahtzee

11111050

Chance

66666030

TotalScore

00001105

BonusUpper

00001035

TotalUpper

00001140

YahtzeeBonus

00001100

TotalLower

00001335

GrandTotal

00001475

NewGame

Aces

66661001

Twos

25555002

Threes

61345003

Fours

12345004

Fives

25222005

Sixes

65432006

ThreeOfKind

11166015

FourOfKind

44443019

FullHouse

66123000

SmallStraight

14563030

LargeStraight

24315040

Yahtzee

66666050

Chance

23456020

TotalScore

00001021

BonusUpper

00001000

TotalUpper

00001021

YahtzeeBonus

00001000

TotalLower

00001174

GrandTotal

00001195

NewGame

Aces

23565!!!

Aces

11116!!!

FullHouse

66666000

//YahtzeeScoreModule.h

#pragma once

#include <string>

using namespace std;

class YahtzeeScoreModule {

private:

public:

YahtzeeScoreModule();

~YahtzeeScoreModule();

int returnScore, acesScore = -1, twosScore = -1, threesScore = -1, foursScore = -1, fivesScore = -1, sixesScore = -1, totalScore = -1, threeOfKindScore = -1,

fourOfKindScore = -1, yahtzeeScore = -1, chanceScore = -1, yahtzeeBonusCount = 0, totalUpper = -1, yahtzeeBonus, totalLower = -1, bonus, grandTotal,

fullHouseScore = -1, smallStraightScore = -1, largeStraightScore = -1;

int aces(int \*rolls);

int twos(int \*rolls);

int threes(int \*rolls);

int fours(int \*rolls);

int fives(int \*rolls);

int sixes(int \*rolls);

int threeOfKind(int \*rolls);

int fourOfKind(int \*rolls);

int yahtzee(int \*rolls);

int chance(int \*rolls);

int fullHouse(int \*rolls);

int smallStraight(int \*rolls);

int largeStraight(int \*rolls);

int functionCaller(int rolls[5], string category);

};

//YahtzeeScoreModule.cpp

#include "YahtzeeScoreModule.h"

#include <iostream>

#include <string>

using namespace std;

YahtzeeScoreModule::YahtzeeScoreModule() {

acesScore = -1, twosScore = -1, threesScore = -1, foursScore = -1, fivesScore = -1, sixesScore = -1, totalScore = -1, threeOfKindScore = -1,

fourOfKindScore = -1, yahtzeeScore = -1, chanceScore = -1, yahtzeeBonusCount = 0, totalUpper = -1, totalLower = -1,

fullHouseScore = -1, smallStraightScore = -1, largeStraightScore = -1;

}

YahtzeeScoreModule::~YahtzeeScoreModule() {

}

int YahtzeeScoreModule::functionCaller(int rolls[5], string category) {

if (\*(rolls) == \*(rolls + 1) && \*(rolls) == \*(rolls + 2) && \*(rolls) == \*(rolls + 3) && \*(rolls) == \*(rolls + 4) && yahtzeeScore == 50) {

yahtzeeBonusCount += 1;

}

if (category == "TotalScore") {

returnScore = totalScore;

}

if (category == "BonusUpper") {

returnScore = bonus;

}

if (category == "TotalUpper") {

returnScore = totalUpper;

}

if (category == "YahtzeeBonus") {

returnScore = yahtzeeBonus;

}

if (category == "TotalLower") {

returnScore = totalLower;

}

if (category == "GrandTotal") {

returnScore = grandTotal;

}

if (category == "Aces") {

returnScore = aces(rolls);

}

if (category == "Twos") {

returnScore = twos(rolls);

}

if (category == "Threes") {

returnScore = threes(rolls);

}

if (category == "Fours") {

returnScore = fours(rolls);

}

if (category == "Fives") {

returnScore = fives(rolls);

}

if (category == "Sixes") {

returnScore = sixes(rolls);

}

if (category == "ThreeOfKind") {

returnScore = threeOfKind(rolls);

}

if (category == "FourOfKind") {

returnScore = fourOfKind(rolls);

}

if (category == "Yahtzee") {

returnScore = yahtzee(rolls);

}

if (category == "Chance") {

returnScore = chance(rolls);

}

if (category == "FullHouse") {

returnScore = fullHouse(rolls);

}

if (category == "SmallStraight") {

returnScore = smallStraight(rolls);

}

if (category == "LargeStraight") {

returnScore = largeStraight(rolls);

}

if (returnScore == -1) {

return returnScore;

}

if (acesScore >= 0 && twosScore >= 0 && threesScore >= 0 && foursScore >= 0 && fivesScore >= 0 && sixesScore >= 0 && totalScore == -1) {

totalScore = acesScore + twosScore + threesScore + foursScore + fivesScore + sixesScore;

if (totalScore >= 63) {

bonus = 35;

}

else bonus = 0;

totalUpper = totalScore + bonus;

}

if (threeOfKindScore >= 0 && fourOfKindScore >= 0 && fullHouseScore >= 0 && smallStraightScore >= 0 && largeStraightScore >= 0 && yahtzeeScore >= 0 && chanceScore >= 0 && totalUpper >= 0) {

yahtzeeBonus = yahtzeeBonusCount \* 100;

totalLower = threeOfKindScore + fourOfKindScore + fullHouseScore + smallStraightScore + chanceScore + largeStraightScore + yahtzeeScore + yahtzeeBonus;

grandTotal = totalUpper + totalLower;

}

return returnScore;

}

int YahtzeeScoreModule::aces(int \*rolls) {

if (acesScore >= 0) {

return -1;

}

acesScore = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

acesScore += 1;

}

}

return acesScore;

}

int YahtzeeScoreModule::twos(int \*rolls) {

if (twosScore >= 0) {

return -1;

}

twosScore = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosScore += 2;

}

}

return twosScore;

}

int YahtzeeScoreModule::threes(int \*rolls) {

if (threesScore >= 0) {

return -1;

}

threesScore = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesScore += 3;

}

}

return threesScore;

}

int YahtzeeScoreModule::fours(int \*rolls) {

if (foursScore >= 0) {

return -1;

}

foursScore = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursScore += 4;

}

}

return foursScore;

}

int YahtzeeScoreModule::fives(int \*rolls) {

if (fivesScore >= 0) {

return -1;

}

fivesScore = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesScore += 5;

}

}

return fivesScore;

}

int YahtzeeScoreModule::sixes(int \*rolls) {

if (sixesScore >= 0) {

return -1;

}

sixesScore = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesScore += 6;

}

}

return sixesScore;

}

int YahtzeeScoreModule::threeOfKind(int \*rolls) {

if (threeOfKindScore >= 0) {

return -1;

}

threeOfKindScore = 0;

int onesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

onesCount++;

}

}

int twosCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosCount++;

}

}

int threesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesCount++;

}

}

int foursCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursCount++;

}

}

int fivesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesCount++;

}

}

int sixesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesCount++;

}

}

if (onesCount < 3 && twosCount < 3 && threesCount < 3 && foursCount < 3 && fivesCount < 3 && sixesCount < 3) {

return 0;

}

for (int i = 0; i < 5; i++) {

threeOfKindScore += \*(rolls + i);

}

return threeOfKindScore;

}

int YahtzeeScoreModule::fourOfKind(int \*rolls) {

if (fourOfKindScore >= 0) {

return -1;

}

fourOfKindScore = 0;

int onesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

onesCount++;

}

}

int twosCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosCount++;

}

}

int threesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesCount++;

}

}

int foursCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursCount++;

}

}

int fivesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesCount++;

}

}

int sixesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesCount++;

}

}

if (onesCount < 4 && twosCount < 4 && threesCount < 4 && foursCount < 4 && fivesCount < 4 && sixesCount < 4) {

return 0;

}

for (int i = 0; i < 5; i++) {

fourOfKindScore += \*(rolls + i);

}

return fourOfKindScore;

}

int YahtzeeScoreModule::yahtzee(int \*rolls) {

if (yahtzeeScore >= 0) {

return -1;

}

yahtzeeScore = 0;

int onesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

onesCount++;

}

}

int twosCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosCount++;

}

}

int threesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesCount++;

}

}

int foursCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursCount++;

}

}

int fivesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesCount++;

}

}

int sixesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesCount++;

}

}

if (onesCount == 5 || twosCount == 5 || threesCount == 5 || foursCount == 5 || fivesCount == 5 || sixesCount == 5) {

yahtzeeScore = 50;

return yahtzeeScore;

}

else return 0;

}

int YahtzeeScoreModule::chance(int \*rolls) {

if (chanceScore >= 0) {

return -1;

}

chanceScore = 0;

for (int i = 0; i < 5; i++) {

chanceScore += \*(rolls + i);

}

return chanceScore;

}

int YahtzeeScoreModule::fullHouse(int \*rolls) {

if (fullHouseScore >= 0) {

return -1;

}

int onesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

onesCount++;

}

}

int twosCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosCount++;

}

}

int threesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesCount++;

}

}

int foursCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursCount++;

}

}

int fivesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesCount++;

}

}

int sixesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesCount++;

}

}

if (onesCount == 2 || twosCount == 2 || threesCount == 2 || foursCount == 2 || fivesCount == 2 || sixesCount == 2) {

if (onesCount == 3 || twosCount == 3 || threesCount == 3 || foursCount == 3 || fivesCount == 3 || sixesCount == 3) {

fullHouseScore = 25;

return 25;

}

else {

fullHouseScore = 0;

return 0;

}

}

fullHouseScore = 0;

return 0;

}

int YahtzeeScoreModule::smallStraight(int \*rolls) {

if (smallStraightScore >= 0) {

return -1;

}

int onesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

onesCount++;

}

}

int twosCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosCount++;

}

}

int threesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesCount++;

}

}

int foursCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursCount++;

}

}

int fivesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesCount++;

}

}

int sixesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesCount++;

}

}

if ((onesCount >= 1 && twosCount >= 1 && threesCount >= 1 && foursCount >= 1) || (twosCount >= 1 && threesCount >= 1 && foursCount >= 1 && fivesCount >= 1) || (threesCount >= 1 && foursCount >= 1 && fivesCount >= 1 && sixesCount >= 1)) {

smallStraightScore = 30;

return 30;

}

else {

smallStraightScore = 0;

return 0;

}

}

int YahtzeeScoreModule::largeStraight(int \*rolls) {

if (largeStraightScore >= 0) {

return -1;

}

int onesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 1) {

onesCount++;

}

}

int twosCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 2) {

twosCount++;

}

}

int threesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 3) {

threesCount++;

}

}

int foursCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 4) {

foursCount++;

}

}

int fivesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 5) {

fivesCount++;

}

}

int sixesCount = 0;

for (int i = 0; i < 5; i++) {

if (\*(rolls + i) == 6) {

sixesCount++;

}

}

if ((onesCount == 1 && twosCount == 1 && threesCount == 1 && foursCount == 1 && fivesCount ==1) || (twosCount == 1 && threesCount == 1 && foursCount == 1 && fivesCount == 1 && sixesCount ==1)) {

largeStraightScore = 40;

return 40;

}

else {

largeStraightScore = 0;

return 0;

}

}