Project 2: Revenge of the Roulette

Curtis Stephens

Csc5-42641

Spring 2017

**Table of Contents**

**Introduction……………………………… 3**

**Summary**

**(And What Has Changed)………… 4**

**Room for Improvement………….. 6**

**Reference Sheet……………………... 7**

**Pseudo Code………………………….... 9**

**Flowchart……………………………….. 15**

**The Program…………………………… 16**

**INTRODUCTION:**

Over the years I have always been attached to the game of roulette. I have spent countless hours (and dollars) trying to figure out different systems that would increase my likelihood of walking out of the casino with more money in my wallet than I walked in with. Video roulette has always been my favorite because not only is it a faster means of game play, but you can also can play more numbers and place more bets with ease. My first attempt at this project fell short on betting systems but I feel this time I around I am much closer to hitting the nail on the head.

For those living under a rock, roulette is a basic guessing game. Traditionally played on a big table where a ball is spun around a wheel until it lands on a number of a certain color grouping, my game aims to achieve this by randomizing numbers and matching the results. To increase the difficulty of game play I made it so you can only guess on 10

numbers per spin, but hitting this number will pay out greatly! My system of betting is much more accurate and realistic than before and makes the gameplay much more interesting!

**Summary (And What Has Changed):**

Project Size: About 500 Lines

Little did you know, but this casino you are playing at is a member’s only casino reserved only for a select few. Upon entry you are asked for your member number and name. If these do not match you are immediately escorted off of the premises. Also since you are member, one of the major perks is receiving a bonus upon return. This bonus is generated through an outside function and then returned and added to your bank. Another new feature is choosing how many chips you would like to buy, but be careful not to break the bank.

As before there are two basic game types. You can play by color (which essentially could have been even or odd as well) or to play by number. The color play is a basic Boolean style game. A user is asked whether he would like to play either Black or Red. The program then generates the right answer between the two and compares it to your guess. Unlike last time though, you are allowed to choose how much you would like to play per spin rather than an automatic amount of $1. Depending on whether you win or lose, your bank is adjusted while factors like numbers of wins, losses, and games are tallied up for statistical purposes. You can end game play at any time by entering -1.

The second (and more complex) game is playing by number. On a true roulette table there are no limits to how many plays you can play per spin. As I mentioned before my game is just a little bit different as I limit the number of plays to make the game more difficult. First the user will enter the number of plays they wish to play on this spin (a value of -1 ends game). A random number between 0 and 37 is generated because there are 38 different places to play on a traditional table. The program then loops the number of times you want to play for this spin. The loops will ask the user to play a number between 0-37, then compares this to the correct answer. If your play is correct your bank will gain 40 times the amount you chose to bet. Wins and losses, just like before, are tallied. If you lose your bank will decrease the amount you chose to bet.

The final part of the game is the information display portion. This was mainly added to fulfill the requirements of this project and has little to do with roulette but still makes the program more appealing. As explained, your wins, losses, and number of game plays are tallied up and you can choose the ratios you would like to see. Also included is a member list function and a function to table your data if you so please.

**Room for Improvement:**

Obviously this game is perfect, but for what is required it is more than complete. On a traditional roulette table you can play chips in between numbers to increase your odds of winning, though it does decrease your winnings. It would be interesting to figure out all the different number combinations and the odds associated with playing in certain places. Also as with my last project I attempted to add the green number function but do to limitations on time I was not able to perfect this concept. Virtualizing the game would make these concepts much easier to work with and as I progress through programming classes I fully intend on updating this project until I get what I consider the perfect form of this game.

**Reference Sheet:**

|  |  |  |  |
| --- | --- | --- | --- |
| Chapter | Section | Topic | Line Number |
| 2 | 2 | Cout |  |
|  | 3 | libraries | iostream,iomanip,cmath,cstdlib,fstream,string,ctime,vec |
|  | 4 | variables/literals | Lines 31-44 |
|  | 5 | Identifiers | Break, const, string, switch, unsigned, static, int, etc |
|  | 6 | Integers | Follow Variables |
|  | 7 | Characters | Lines 38, 62, 122 |
|  | 8 | Strings | Lines 37, 54, 58, 225, 275, 288 |
|  | 9 | Floats No Doubles | Reference variables list |
|  | 10 | Bools | Lines 61, 70, 359, 409 |
|  | 11 | Sizeof \*\*\*\*\* | Achieved! |
|  | 12 | Variables 7 characters or less | Achieved! |
|  | 13 | Scope \*\*\*\*\* No Global Variables | Achieved! |
|  | 14 | Arithmetic operators | + - \* / % |
|  | 15 | Comments 20%+ | Achieved! |
|  | 16 | Named Constants | Lines 33-36 and also for array size |
|  | 17 | Programming Style \*\*\*\*\* Emulate | Achieved! |
| 3 | 1 | cin | Throughout project |
|  | 2 | Math Expression | Throughout project |
|  | 3 | Mixing data types \*\*\*\* | Achieved! |
|  | 4 | Overflow/Underflow \*\*\*\* |  |
|  | 5 | Type Casting | Line 187 |
|  | 6 | Multiple assignment \*\*\*\*\* | Lines 155 |
|  | 7 | Formatting output | Lines 192 |
|  | 8 | Strings | Lines 37, 44, 54, 58, 225, 275, 288 |
|  | 9 | Math Library | Line 13 |
|  | 10 | Hand tracing | \*\*\*\*\*\* |
| 4 | 1 | Relational Operators | Lines 62, 68, 79, 87, 103, 112, 118, 122, 131, 140, etc |
|  | 2 | if | Lines 62, 79, 152 |
|  | 4 | If-else | Lines 95, 162, 183 |
|  | 5 | Nesting | Used throughout project |
|  | 6 | If-else-if | Line 122 |
|  | 7 | Flags \*\*\*\*\* | Line 183 |
|  | 8 | Logical operators | Lines 62, 68, 112, 122, 145 |
|  | 11 | Validating user input | Lines 68, 112, 131, 145, 183, 217 |
|  | 13 | Conditional Operator | Lines 68, 112, 131, 145 |
|  | 14 | Switch | Lines 280-318 |
| 5 | 1 | Increment/Decrement | Lines 82-84, 98-100, 155-157, 165-167 |
|  | 2 | While | Used for validating input |
|  | 5 | Do-while | Lines 74-118, 126-179, |
|  | 6 | For loop | Lines 140-176 |
|  | 11 | Files input/output both | Lines 47-51, 223-230 |
|  | 12 | No breaks in loops \*\*\*\*\*\* | Achieved! |
| 6 | 3 | Function Prototypes | Lines 326-336, 338-349, 351-370, 373-397, 399-483 |
|  | 5 | Passing by value | Lines 42 & 344 |
|  | 8 | Returning values from functions | Lines 334, 348, 369, 423, 433 |
|  | 9 | Returning a boolean \*\*\*\*\*\* | Position returns for search |
|  | 10 | No Global Variables Allowed | Achieved! |
|  | 11 | Static Local | Lines 426-434 |
|  | 12 | Default arguments | Lines 31 & 436-454 |
|  | 13 | Reference Parameters | Lines 26 &342 |
|  | 14 | Overloading functions | Lines 25, 26, 342, 355 |
|  | 15 | Exit function \*\*\*\*\*\*\* |  |
| 7 | 4 | Array Initialization | Lines 52-58, 443-449 |
|  | 6 | Processing Arrays | Lines 471, 474 |
|  | 7 | Parallel Arrays | Lines 436-454 |
|  | 8 | Arrays as function arguments | Lines 377, 403 |
|  | 9 | 2-D Arrays | Lines 456-483 |
|  | 12 | STL Vector | Lines 51, 169-170 |
| 8 | 1 | Linear and Binary Search | Lines 351-370, 399-424 |
|  | 3 | Bubble and Selection Sort |  |
|  | 5 | Search/Sorting Vectors \*\*\*\*\*\* | Lines 373-397 |
|  |  |  |  |
|  |  |  |  |

**Pseudo Code:**

/\*

File: main.cpp

Author: Curtis Stephens

Created in 2017

Purpose: Pseudo code for roulette game \*/

//System Libraries //Input Output library //Random numbers //Time to set the Seed //String Functioning //Math Library //Precision library //Read Write Library //File stream library //Format Library

//Vector Library

//Namespace std of system libraries

//User Libraries //Global Constants

//Such as PI, Vc, -> Math/Science values

//as well as conversions from system of units to another //Percentage Conversion

//Function Prototypes

//Main -> Executable code begins here!!!

//Declare Variables and Initialize

//int for guesses, floats for ratios //char and string for choices

//Set Constants for limits, Set size

//Counters/indicators initialize for wins, loss, and $

//Intialize arrays and set arrays for valid accounts and names

//Intro

//Instantiate and Open files for header //Retrieve and Display Header

//Close file

//Input Account Number

//Linear Search function

//Input name

//Binary Search Function

//Bank and Choice

//Enter Bank Amount

//Call Bonus Function

//Choose game

//Play By Color

//Ask User to Bet on Black or Red //Validate Input

//Play by Do While Looping

//Call Random Function

//Compare Choice

//If Win

//Display Winning Message

//Add Bet to Bank

//Add 1 to Win Tally

//Add 1 to Games Tally

//Display Bank Total

//Else Lose

//Display Losing Message

//Subtract Bet From Bank

//Add 1 to Loss Tally

//Add 1 to Games Tally

//Display Bank Total

//If Money < 0

//Display Bankrupt Message

//Break! End Game

//Play Another Game.

//Validate Input

//Do While Loop Ends Game With -1

//Play By Number

//Ask User to How Many Plays on This Spin

//Validate Input

//Play by Do While Looping

//Ask for Number of Plays on This Spin

/Validate Input

//Call Random Number Function.

//Add random number to vector

//Initiate For Loop for Number Guess

//Choose Number

//Validate Input

//Compare Choice To Random Number

//If Win

//Display Win Message

//Bet\*40 is win. Add to Bank

//Add $40 to Bank

//Add 1 to Play Tally

//Add 1 to Win

//Display Bank Total

//Else Lose

//Display Loss Message

//Subtract Bet from Bank

//Add 1 To Loss Tally

//Add 1 to Play Tally

//If Money < 0

//Display Bankrupt Message

//Break! End Game

//Display Winning Number

//Do While Loop Ends With -1 Entry

//Ratios

//Algebraic and Static Expressions for Answer

//Output Data

//Set Precision for floats

//Display Win Total

//Display Loss Total

//Ask For Ratio Display

//Switch Menu for Ratio Display

//W Displays Win vs Plays

//L Displays Loss vs Plays

//O Displays Win Over Loss

//T Displays All Three

//Display Ending Bank Balance

//Member List

//Ask if they want to see member list

//If Yes

//Call Member List Function

//Display Member List

//If No

//Alright Meesage

//Winner List

//Ask if they want to see winning number list

//If yes

//Open Output File

//For Loop While Index is < plays

//Display Vector showWinner[index]

//Output number to output file

//Index++

//Close outputFile

//If No

//Alright Message

//Table

//Ask if they want to create a table

//If Yes

//Call table Function

//Display Data

//If No

//Alright Message

//Write File

//Open Output File

//Output Win Total In Output File

//Output Loss Total In Output File

//Output Win Ratio In Output File

//Output Loss Ratio In Output File

//Output Win/Loss In Output File

//Output End Balance In Output File

//Close File

//Exit!!!

//Functions

//Random Number for Colors Function

//Set Seed

//Randomize Between Black (1) and Red (2)

//Return Results

//Random Number for Numbers Function

//Set Seed

//Randomize Numbers Between 0 and 37

//Return Results

//Linear Search for Account Numbers

//Pull Valid Account Array, Size, And Account Inputed

//Declare Variables

//Search Array and Compare to Input

//Return Results

//Sort Names Function

//Pull Names Array and Size

//Declare Values

//Sort Names

//Return Array

//Bank Bonus Function

//Pull Bank

//Add 5

//Return Value

//Show List Parallel Function

//Declare Array Size

//Declare Names Array and Account Number Array

//Display List

//2d Array To Create Table

//Declare rows, colums for size

//Initiate Array for Table

//Input Id, Starting Amount, Bonus, End Total

//Calculate Missing Parts

//Create Table

**Flowchart:**

Due to formatting issues I was not able to fit my flowchart in a way that made it logical. Please see the following link to view the flowchart for the game.

**The Program:**

/\*

\* File: main.cpp

\* Author: Curtis Stephens

\* Created in 2017

\* Purpose: Final Project Roulette Game

\*/

//System Libraries

#include <iostream> //Input - Output Library

#include <ctime> //Time for rand

#include <cstdlib> //Srand to set the seed

#include <string> ///string function

#include <cmath> // math functions

#include <iomanip> //set precision

#include <fstream> //read file

#include <vector> //vector

using namespace std; //Name-space under which system libraries exist

//User Libraries

//NO CONSTANTS!!!!!

//Function Prototypes

int random(int, int); //color random

long random(int &); //number random

int searchList(int[],int,int); //search linear

void selectionSort(string [], int); //sort

int binarySearch(string[], int, string); //binary search

int bank(int, int); //multiply bank

void showList(int = 0); //parrallel arrays

void twoDee(); //2d array

int bonus(); //bonus function

//Execution begins here

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

//Declare variables

string choice, name, title;

char winLose; //winlose

int color, guessMax, guessed, winner2, moneys; //random variables

int black = 1; //black

int red = 2; //red

int min=0; //minimum for number game

int max=10; //max that gets changed

unsigned int money; //starting bank

int bet; //betting amount

float wins = 0; //set win counter

float losses = 0; //set lose counter

float winRatio, loseRatio, winLost; //ratios

float plays = 0;//set play counter

int results, acctNum; //verify acct

const int acctSize=20; //account size

vector<int>storeWinner;

int validAcct[acctSize]={543, 785, 313, 432, 765, 345, 678, 409, 945,

284, 851, 274, 345, 456, 235, 143, 178, 993,

169, 420}; //valid accts

string names[acctSize]={"Mark", "David", "Henry", "Kristina", "Joel",

"Herman", "Tyler", "Kohl", "Tracy", "Coby", "Nina",

"Sean", "John", "Mia", "Thomas", "Nick", "Pat",

"Eric", "Robert", "Curtis"}; //valid names

//Intro

ifstream inputFile; //input file

inputFile.open("Title.txt"); //source file

inputFile >> title; //input info

cout << title << "!!" << endl; //display info

inputFile.close(); //close file

cout<<"Welcome to the Roulette Game!\n"; //greeting

cout<<"Please verify your account.\n";

cout<<"Enter your three digit account number (for prof. access use 543): ";

cin>>acctNum; //enter account number

//validate account

results=searchList(validAcct, acctSize, acctNum); //call search function

if (results==-1) //invalid account number

{

cout<<"The account is not valid.\n";

}

else //valid account number

{

cout<<"For verification what is your name (prof. access use Mark)? ";

cin>>name; //enter name

//validate name

selectionSort(names,acctSize); //sort function

int results=binarySearch(names, acctSize, name); //binary search function

if (results==-1) //invalid account

{

cout<<"The name was not found\n";

}

else //valid account play game

{

//initiate bank

cout<<"Welcome back player number "<<acctNum<<"!\n";

cout<<"How many dollars in chips would you like to purchase? $";

cin>>money;

int bonused=bonus();

cout<<"You recieved a $"<<bonused<<" bonus for your return!\n";

moneys=bank(money, bonused);

cout<<"Your bank is $"<<moneys<<endl;

cout << "Play by color (C) or play by number (N)? ";

cin >> choice; //choose game type

cout << "\n";

//play

if (choice == "C" || choice == "c") //choose the color game

{

cout<<"You chose to play by number!\n";

cout << "Choose a color! Black (1) or Red (2). To quit enter (-1): ";

cin >> color; //choose color

do

{

while (color < -1 || color > 2) //validate input

{

cout << "Not a valid choice! Choose either Black (1) or ";

cout << "Red (2): ";

cin >> color;

}

int winner1=random(red, black);

//win!!!!

cout<<"How much do you want to bet on this spin? $";

cin>>bet;

while(bet>40||bet<1)

{

cout<<"Cannot bet more than $40. Reenter bet: $";

cin>>bet;

}

if (color == winner1)

{

cout << "You win!" << endl;

moneys+=bet; //adjust bank

wins++; //add win

plays++; //add play

cout << "You have $" << moneys;

cout << " left. \n" << endl; //funds left

cout << endl;

}

//lose!!!

else

{

cout << "You lose." << endl;

moneys-=bet; //adjust bank

losses++; //add loss

plays++; //add play

cout << "You have $" << moneys;

cout << " left.\n" << endl; //funds left

if (moneys <= 0) //bankrupt

{

cout << "You are out of money!! Come Again!" << endl;

break; //end game from lack of funds

}

}

cout << endl;

cout << "Enter another color. Black (1) or Red (2)(-1 to quit): ";

cin >> color; //repeat loop

}while (color != -1); // ends games

}

else if (choice == "N" || choice == "n") //choose number game

{

do //start number game

{

cout << "How many plays do you want for this spin? (up to 10 per ";

cout << "spin enter -1 when done): ";

cin >> guessMax; //how many guesses for this spin

while (guessMax < -1 || guessMax > 10) //validate

{

cout << "Not a valid number of plays! Reenter ";

cout << "number of plays (1-10): ";

cin >> guessMax; //reenter

}

//generate winning number

int winner2=random(min);

storeWinner.push\_back(winner2);

//guess and compare

for (int guess = 1; guess <= guessMax; guess++) //guess loop

{

cout << "What number would you like to play for ";

cout << "play number " << guess << "? (0-37): ";

cin >> guessed; //guess number

cout<<"How much would you like to bet for this play? $";

cin>>bet; //enter bet

int winAmounted = bet\*40; //win amount

while (guessed < 0 || guessed > 37) //validate input

{

cout << "Not a valid number to play! Please reenter ";

cout << "The number that you want to play (0-37): ";

cin >> guessed;

}

//win

if (guessed == winner2) //winning number

{

cout << "This number hit!" << endl; //win

moneys += winAmounted; //add winnings

plays++; //add play

wins++; //add wins

cout << "You have $" << moneys;

cout << " left. \n" << endl; //funds left

}

//lose

else // losing number

{

cout << "This number missed." << endl; //lose

moneys-=bet; //subtract losings

losses++; //add losses

plays++; //add plays

cout << "You have $" << moneys;

cout << " left.\n" << endl; //funds left

if (moneys <= 0) //bankrupt!!

{

cout << "You are out of money!! Come Again!" << endl;

break;

}

}

}

cout << "The Winning Number is ";

cout << winner2 << "\n\n"; //display winning number

} while (guessMax != -1); //to end game

}

else

cout << "Not a valid game choice. Please come again!";

}

//Output the transformed data

//ratios

winRatio = static\_cast<double>(wins) / plays; //static

loseRatio = (losses / plays); //figure out ratios

winLost = (wins / losses);

//output data

cout << "Your total number of wins is " << wins << endl; //total wins

cout << "Your total number of losses is ";

cout << losses << endl << endl;//total losses

cout << setprecision(2) << fixed;

cout << "Would you like to see Win ratio (W), ";

cout << "Loss ratio (L), Win over Loss ratio (O), ";

cout << "or All Three (T)?: ";

cin >> winLose; //choose ratio

switch (winLose) //switch option for ratios

{

case 'W': //win ratio

case 'w': cout << "Your Win ratio is " << winRatio << endl;

break;

case 'L': //loss ratio

case 'l': cout << "Your Loss ratio is " << loseRatio << endl;

break;

case 'o': //win loss ratio

case 'O': cout << "Your Win over Loss ratio is " << winLost << endl;

break;

case 'T':

case 't': //display all three

cout << "Your Win ratio is " << winRatio << endl;

cout << "Your Loss ratio is " << loseRatio << endl;

cout << "Your Win/Loss ratio is " << winLost << endl;

break;

default: cout << "That is an invalid choice. \n"; //validation

}

cout << "Your ending balance is $" << moneys << endl << endl; //end balance

//member list

cout<<"Would you like to see the current member list(Y for yes)? ";

string listChoice;

cin>>listChoice; //choose list

if(listChoice=="Y"||listChoice=="y") //yes want list

{

cout<<"Member List\n";

cout<<"------------------\n";

showList();

cout<<endl;

}

else //dont want to

{

cout<<"Alright.\n";

}

//Winner List

cout<<"Would you like to view the winning numbers(Y for yes)? ";

string winnerList;

cin>>winnerList; //choose to see list

if(winnerList=="Y"||winnerList=="y") //yes want to see

{

ofstream outputFile; //open file

outputFile.open("Winners.txt");

for(int index=0;index<plays;index++) //for loop to display winning

{

cout<<storeWinner[index]<<" "; //display

outputFile<<storeWinner[index]<<" "; //write winning number to file

}

outputFile.close(); //close file

cout<<endl;

}

else //Dont want to

{

cout<<"Alright.\n";

}

//Table the Data

cout<<"Would you like to table the data(Y for yes)? ";

string tableChoice;

cin>>tableChoice; //choose

if(tableChoice=="Y"||tableChoice=="y") //yes want to table

{

twoDee(); //2d function

cout<<endl;

}

else //dont want to

{

cout<<"Alright.\n";

}

//write file

ofstream outputFile; //open file

outputFile.open("Output.txt");

outputFile << "Player: " << name << endl; //write name to file

outputFile << "Your number of Wins is " << wins << endl; //write wins

outputFile << "Your number of Losses is " << losses << endl; //write losses

outputFile << "Your Win ratio is " << winRatio << endl; //write win ratio

outputFile << "Your Loss ratio is " << loseRatio << endl; //write lose

outputFile << "Your Win/Loss ratio is " << winLost << endl;//write ratio

outputFile << "Your ending balance is $" << money << endl; //write balance

outputFile.close(); //close file

//Exit stage right!

}

return 0;} //end

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

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

//randomize numbers color

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

int random(int red, int black)

{

unsigned seed = time(0); //set seed

srand(seed); //initialize random

int winner1=(rand() % (red - black + 1)) + black; //randomize

return winner1; //return

}

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

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

//randomize numbers number

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

long random(int &min)

{

int max=37; //set max

unsigned seed = time(0); //set seed

srand(seed); //intialize random

int winner2=(rand() % (max - min + 1)) + min; //randomize

return winner2; //return random

}

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

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

//search linear

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

int searchList(int validAcct[], int acctSize, int acctNum)

{

int index=0; //variables

int position=-1;

bool found=false;

while(index<acctSize&&!found) //search

{

if(validAcct[index]==acctNum)

{

found=true;

position=index;

}

index++;

}

return position; //return results

}

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

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

//sort names

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

void selectionSort(string names[], int acctSize)

{

int startScan, minIndex; //variables

string minValue;

for(startScan = 0; startScan < (acctSize-1); startScan++) //sort

{

minIndex = startScan;

minValue = names[startScan];

for(int index = startScan +1; index <acctSize; index++)

{

if (names[index] < minValue)

{

minValue = names[index];

minIndex = index;

}

}

names[minIndex] = names[startScan];

names[startScan] = minValue;

}

}

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

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

//search names

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

int binarySearch(string names[], int acctSize, string name)

{

long first=0; //variables

long last=acctSize-1;

long middle;

int position=-1;

bool found=false;

while(!found&&first<=last) //search binary

{

middle=(first+last)/2; //startys middle

if(names[middle]==name)

{

found=true;

position=middle;

}

else if (names[middle]>name)

last=middle-1;

else

first=middle+1;

}

return position; //return position

}

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

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

//bank bonus static local

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

int bank(int money, int bonus)

{

static int moneys=(money+bonus); //add bonus to money

return moneys; //return money

}

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

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

//parallel arrays

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

void showList(int index)

{

const int size=20; //variables size

int validAcct[size]={543, 785, 313, 432, 765, 345, 678, 409, 945,

284, 851, 274, 345, 456, 235, 143, 178, 993,

169, 420}; //valid accts

string names[size]={"Mark", "David", "Henry", "Kristina", "Joel",

"Herman", "Tyler", "Kohl", "Tracy", "Coby", "Nina",

"Sean", "John", "Mia", "Thomas", "Nick", "Pat",

"Eric", "Robert", "Curtis"}; //valid names

for(index=0;index<size;index++) //for loop to display list

{

cout<<validAcct[index]<<" "<<names[index]<<endl; //display

}

}

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

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

//2d array

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

void twoDee()

{

const int rows=3;

const int cols=3;

int entries[rows][cols];

cout<<"Enter your id number: ";

cin>>entries[0][0]; //enter id

cout<<"What was your starting amount? $";

cin>>entries[1][1]; //enter starting amount

cout<<"Did you have any bonuses? $";

cin>>entries[1][2]; //enter bonus amount

entries[1][3]=(entries[1][1]+entries[1][2]); //bank total

cout<<"What was your ending total? $";

cin>>entries[2][1]; //end total

entries[3][1]=entries[1][3]-entries[2][1]; //difference

cout<<"Table\n"; //display table

cout<<"-------------------\n";

cout<<"Id #:"<<entries[0][0]<<" | Start | Bonus | Totals"<<endl;

cout<<" "<<entries[1][1]<<" "<<entries[1][2];

cout<<" "<<entries[1][3]<<endl;

cout<<" Ending Bank | "<<entries[2][1]<<endl;

cout<<" Difference | "<<entries[3][1]<<endl<<endl;

cout<<"Thank you for coming! Come again!\n";

}

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

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

//randomize bonus

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

int bonus()

{

int min=0;

int max=20; //set max

unsigned seed = time(0); //set seed

srand(seed); //intialize random

int bonused=(rand() % (max - min + 1)) + min; //randomize

return bonused; //return random

}