Andrew Tran

Lab Experience One

**Exercise One:**

You are to do lab1A from your lab manual.

**Exercise Two:**

Write a program that simulates the rolling of two six sided die. The program should call a function called roll that returns the value of the die. This function should use the pseudorandom number generator function, rand().

The sum of the two values should then be calculated. Since each die is six-sided, the sum should range in values from 2 to 12, with 7 being the most frequent and 2 and 12 the least frequent. Note: The sample size (a set of all possible outcomes) is of size 36. I.e. there are six ways to roll a 7, therefore the probability of a 7 should be 6/36 = 1/6. There is one way to roll a two which means the probability is 1/36. Therefore **you will need to generate two random numbers from 1 to 6 to represent the two dies.**

Your program should roll the two die 36,000 times.

Use a two dimensional array to tally the number of times each possible sum appears and print the results in tabular format.

Are the results reasonable?

**How to generate random numbers?**

To generate random numbers in C++ you must include the header file cstdlib, seed the random number generator, and use the rand() function to generate an integer from 0 to MAXINT (a predefined constant).The following code skeleton is used to generate random numbers:

#include <iostream>

#include <ctime>

#include <cstdlib>

int main(){

int number;

srand(time(NULL)); // seed the random number generator with the system clock.

number = 1 + rand() % 6; // generate a number from 1 to 6.

return 0;

}// end main

For full credit your program must use functional decomposition to solve the problem. I.e. you must use more than one function to solve the problem and a two dimensional array to keep a tally of the results. To pass a two dimensional array to a function you must specify the number of columns in the array. The main function does not count as one of the functions of your program.

/\*--- search.cpp -----------------------------------------------------

Program to read a 3 X 3 matrix of integers mat and an integer item,

and search mat to see if it contains item.

Andrew Tran

--------------------------------------------------------------------\*/

#include <iostream>

using namespace std;

const int SIZE = 3; // Set matrix size

typedef int Matrix[SIZE][SIZE]; // Define data type Matrix

bool matrixSearch(Matrix & mat, int n, int item);

/\*---------------------------------------------------------------------

Search the n X n matrix mat in rowwise order for item.

Precondition: Matrix mat is an n X n matrix of integers with n > 0.

Postcondition: True is returned if item is found in mat, else false.

---------------------------------------------------------------------\*/

int main()

{

// Enter the matrix

Matrix mat;

cout << "Enter the elements of the " << SIZE << " X " << SIZE

<< " matrix rowwise:\n";

for (int i = 0; i < SIZE; i++)

for (int j = 0; j < SIZE; j++)

cin >> mat[i][j];

// Search mat for various items

int itemToFind;

char response;

do

{

cout << "Enter integer to search for: ";

cin >> itemToFind;

if (matrixSearch(mat, SIZE, itemToFind))

cout << "item found\n";

else

cout << "item not found\n";

cout << "\nMore items to search for (Y or N)? ";

cin >> response;

}

while (response == 'Y' || response == 'y');

}

//-- (-- Incorrect --) Definition of matrixSearch()

bool matrixSearch(Matrix & mat, int n, int item)

/\*--------------------------------------------------------------------

Search the n X n matrix mat in rowwise order for item

Precondition: Matrix mat is an n X n matrix of integers with n > 0.

Postcondition: True is returned if item is found in mat, else false.

NOTE: mat[row][col] denotes the entry of the matrix in the

(horizontal) row numbered row (counting from 0) and the

(vertical) column numbered col.

-------------------------------------------------------------------\*/

{

bool found=false;

for (int row = 0; row < n; row++)

for (int col = 0; col < n; col++)

{

if (mat[row][col] == item)

{

found = true;

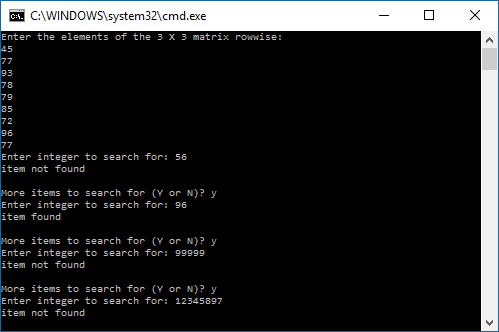
break;

}

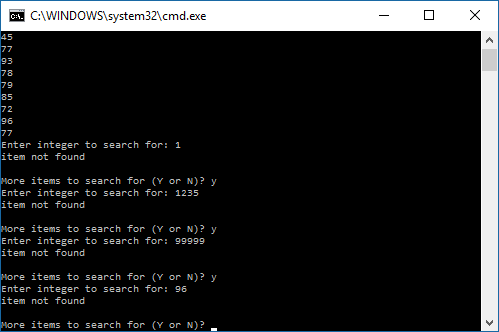
}

return found;

}



(screenshot of original function testing below)



PROBLEM 2:

#include <iostream>

#include <iomanip>

#include <ctime>

#include <cstdlib>

using namespace std;

int diceroll();

void format();

int main()

{

srand(time(NULL)); //seed

int counter[6][6] = { 0 }; //intialize two dimensional array

int result[13] = { 0 };

int die1, die2;

int u = 1, x=0;

for (int roll = 0; roll <= 36000; roll++) //generate random rolls

{

die1 = diceroll();

die2 = diceroll();

counter[die1 - 1][die2 - 1]++;

result[die1 + die2]++;

}

format();

for (int i=0; i<6; i++) //print array

{

cout << u; //formatting

u++; //column of 1-6 index

cout << setw(7); //formatting

for (int j=0; j < 6; j++)

{

cout << counter[i][j] << setw(7); //array print

}

cout << endl;

}

cout << endl;

cout << "Sum" << setw(13) << "Frequency" << endl;

for (int k = 2; k<13; k++)

{

cout << setw(7) << k << setw(13) << result[k] << endl;

}

return 0;

}

int diceroll() //rolls the dice, da dice run hot!

{

return rand() % 6 + 1;

}

void format()

{

cout << " "; //formatting block

for (int y = 1; y <= 6; y++) //row of 1-6 index

{ //

cout << setw(7) << y; //

} //

cout << endl; //

cout << setw(7); //formatting block

}

