CS 1428
Fall 2018
Gentry Atkinson
Lab 4

Introduction:

Up to now we've been working with single value variables. So one variable only represents one number or one character or one boolean value. But sometimes we will want to represent collections of values. For instance, we might want to store the grades of 20 students rather than just one student. Instead of declaring twenty different integer variable, we can declare one integer array.

An array is like a matrix in algebra. It is many different, related values which are all represented as being part of the same collection. Additionally, all of these values will be stored in the same block of memory. In C++ arrays are always of a fixed size and you must tell the compiler what that size is when you declare the array. This can be done in two fashions:

```
int myFirstArray [] = {1, 2, 3, 4};      //providing a list of values to create a size 4 array
int mySecondArray [4];                   //providing a size without values
```

The values stored in an array can be retrieved with with an **index which starts from 0** and goes up to size-1. Retrieving a value from an array looks like this:

```
cout <<  myFirstArray[0];        //prints 1
cout << myFirstArray[3];         //prints 4
```

The purpose of this lab is to familiarize you with the techniques of safely creating and using arrays.

Directions:

1- Launch Code::Blocks and start a new file. Name it your_last_name_lab3.cpp.

2- Include the standard header for this lab:

```
//Your Name
//CS1428 Fall 2018
//Lab 4
```

3- Include the iostream standard library and start your main function:

```
#include <iostream>
using namespace std;
int main() {
```

4- Copy the following code to create, load, and print a simple array

```
int grades[3];

for(int i = 0; i < 3; i++){
        cout << "Enter a grade for student " << i << ": ";
        cin >> grades[i];
}

for(int i = 0; i < 3; i++){
        cout << "Student " << i << " earned a " << grades[i] << endl;
}
```

Notice that the first student is Student 0 because **the first index in grades is 0** and the last student is Student 2 because **the last index in grades is 2.**

5- Rewrite the body of the second **for** loop in problem 4 so that it shows the student's letter grade rather than their number grade. So it should print "Student 0 earned a A" **if** that students score is between 90 and 100, B **if** their score is between 80 and 89, C **if** their score is between 75 and 79, D **if** their score is between 70 and 74, or **else** an F.

6- Your client is a property manager that oversees 8 different apartment complexes. Each complex was recently inspected and given a score of "F" for failing, "P" for passing, or "E" for excellent. The scores assigned to the complexes were: E, P, F, F, E, P, E, F. They need you to:
   1. Write a loop which will count the number of excellent properties and print the result.
   2. Write a loop which will print the index of every failing property.
   3. Update the third property from failing to passing and the seventh property from excellent to passing.

7- Copy the following poorly written code. Debug it so that each line performs the action described in the comment:

```
//declare an array of prime numbers
int primes = {2, 3, 5, 7, 11, 13,  17}

//declare an integer named index
int index;

//declare a float named average
flot average;

//loop while index is less than 7
while index <= 7{
```

//add the value in primes at index to average
average = average + primes;
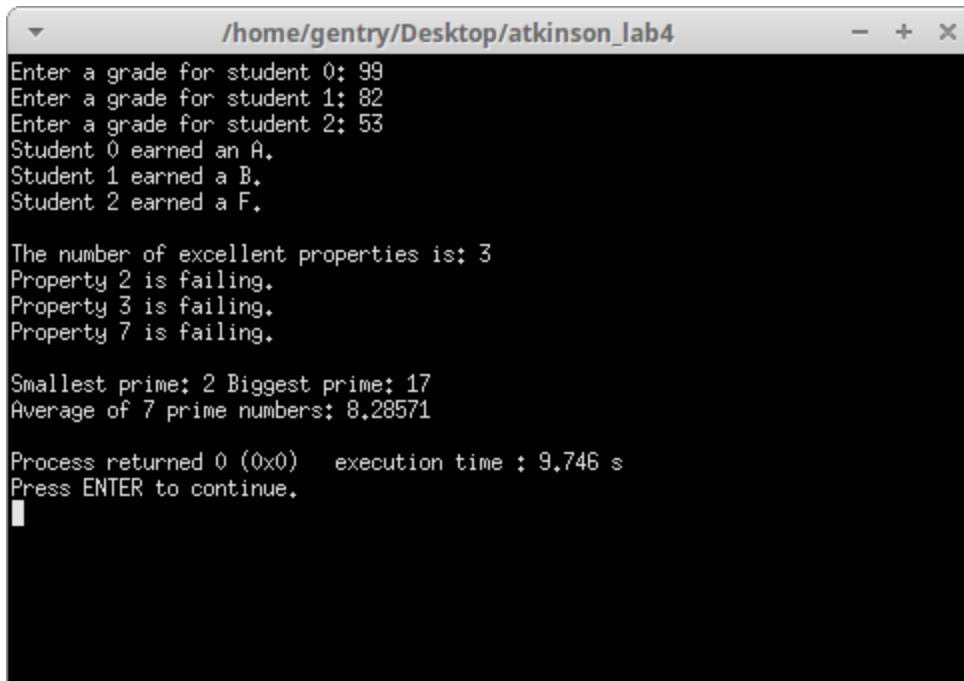
//update index
index+;

}

//print range of averages
cout << "Smallest prime: " << primes[1] << " Biggest prime: " << primes[7] << endl;

//divide average by the count of values in primes
average/7;

//print average
cout << "Average of 7 prime numbers: " << average >> endl;

8- Build and Run your code. Fix any errors. Your output should look something like this:

```
/home/gentry/Desktop/atkinson_lab4                    — + x

Enter a grade for student 0: 99
Enter a grade for student 1: 82
Enter a grade for student 2: 53
Student 0 earned an A.
Student 1 earned a B.
Student 2 earned a F.

The number of excellent properties is: 3
Property 2 is failing.
Property 3 is failing.
Property 7 is failing.

Smallest prime: 2 Biggest prime: 17
Average of 7 prime numbers: 8.28571

Process returned 0 (0x0)   execution time : 9.746 s
Press ENTER to continue.
```

9- Submit your .cpp file through TRACS.