CSCI 1300 – Introduction to Computer Programming
Instructor: Hoenigman/Lewis
Assignment 8
Due Friday, November 13 by 8am

For this assignment, include all of your code in one file, called main.cpp, which is the default source file name in a CodeBlocks project.

Submitting Your Code to Moodle
You must submit your code to Moodle to get full credit for the assignment, even if the computer science autograder gives you a perfect score.

Please also include comments in your code to describe what your code is doing. Comments should also include your name, recitation TA, and the assignment and problem number. TAs will be checking that you code has comments.

Submitting Your Code to the Autograder:
The computer science autograder, known as COG, can be found here:
https://web-cog-csci1300.cs.colorado.edu

Login to COG using your identikey and password. Select the CSCI1300 - Assignment #08 from the dropdown. Upload your .cpp file and click Submit. **Your file needs to be named main.cpp for the grading script to run.** COG will run its tests and display the results in the window below the Submit button. If your code doesn't run correctly on COG, read the error messages carefully, correct the mistakes in your code, and upload a new file. You can modify your code and resubmit as many times as you need to, up until the assignment due date.

Before you submit your code to COG, make sure it runs on your computer. If it doesn't run on the VM, it won't run on COG.

If you do not get your assignment to run on COG before the assignment deadline, you will have the option of scheduling an interview grade with your TA to get a grade for the assignment. We'll talk more about scheduling the interview in lecture and recitation. Even if you do get the assignment to run on COG, you can schedule the interview if you just want to talk about the assignment and get feedback on your implementation.

What to do if you have questions
There are several ways to get help on assignments in 1300, and depending on your question, some sources are better than others. There is a Peer Discussion Forum on our Moodle page that is a good place to post technical questions, such as how to get user input, or treat that input as an integer. When you answer other students' questions on the forum, please do not post entire assignment solutions. The CAs are also a good source of technical information. If, after reading the assignment write-up, you need clarification on what you're being asked to do in the assignment, the

TAs and the course instructors are better sources of information than the discussion forum or the CAs.

The apple-management system

A local apple farmer has asked you to write a program to help him manage his apple inventory. The farmer needs to keep track of how many apples he sells everyday, how many apples he picks, and how many apples he has in storage. Your apple inventory system needs to track data for 30 days.

Like any self-respecting apple farmer, this farmer thinks the inventory data should be stored in C++ arrays. Your program needs two arrays: one array to store the apple harvest for 30 days and one array to store the apple sales for 30 days.

There are six functions that your program needs to implement. Two of the functions, *sellApples()* and *harvestApples()* are used to update the sales and harvest arrays, respectively.

bool sellApples(int sales[], int inventory, int demand, int currentDay)
//update sales[] to set sales[currentDay] = demand
//The sales[] array should be updated with demand if demand <= inventory
//if demand > inventory, then set sales[currentDay] = 0
//if the array is updated with demand, the function should return True.
//Otherwise, the function should return False

void harvestApples(int harvest[], int currentDay, int dayHarvest)
//update harvest[] to set harvest[currentDay] = dayHarvest

Your program also needs a function to check whether there is room available in the *harvest* and *sales* arrays. After 30 days, you can stop storing data and print a nice message to let the user know to stop harvesting and selling apples. The function *endOfMonth()* checks if the array is full by checking if currentDay = 30.

bool endOfMonth(int currentDay)
//check if the array is full by checking if currentDay = 30.
//if the array is full, then return True. Otherwise, return False

Your program also needs function to update the *currentDay* and *inventory* variables.

int updateCurrentDay(int currentDay)
//this function adds 1 to currentDay and returns this value
//return currentDay+1;
//Note: if you use currentDay++ here it won't work.

int updateInventory(int inventory, int change)
/*
Use this function to update the inventory variable after calling *sellApples()*

or *harvestApples()* to change the value of *inventory*. The parameter *change* is the change to inventory. If *change < 0*, it means that apples have been sold, and if *change > 0*, it means that apples have been harvested.

This function should only be called after *sellApples()* is called if *sellApples()* returns True, indicating that the sale was successful and apples need to be deducted from the inventory. The function should also be called after *harvestApples()* to add apples to the inventory.

The function should return the new value for *inventory*.
*/

Once 30 days of data has been collected, calculate the average harvest for the 30 days and the average sales for the 30 days. Your program needs to implement a function for each of these calculations.

double calculateAverageHarvest(int harvest[ ])
//calculate the average daily harvest after 30 days

double calculateAverageSales(int sales[ ])
//calculate the average apple sales after 30 days

Testing your program
The COG autograder will be testing your functions for this assignment, which means it's important that your functions are named exactly as they're shown here. It also means that any printing you do in your main function will be ignored by COG. Any printing inside your functions will confuse COG.

In your main function, you can test that your functions are working correctly by implementing the algorithm shown below. It's important to note that COG won't run your main function for this assignment, any code you write in main will be only for testing your functions. You are welcome to write additional tests in main to verify that your functions are working. For example, you may want to write a while loop that calls *harvestApples()* until *endOfMonth()* is true and update the inventory each time through the loop.

```
Create integer arrays for sales and harvest to store 30 integers
Initialize the current day to 0
//We start with a current day = 0 because it's the index in the array
//The 30 days of data will be stored in the array at indices 0 … 29.

Initialize the inventory to 0

While current day < 30:
      Ask the user for a harvest amount, or read from another array
      Call harvestApples to update the harvest variable
      Update the inventory
      Ask the user for a sales amount, or read from another array
```

```
        Call sellApples to update the sales
        If sellApples returns True
              Update the inventory
        Update the current day
        Check if it’s the end of the month

  Print a nice message saying that you have 30 days of data
  Calculate the average harvest
  Print the average harvest result
  Calculate the average sales
  Print the average sales result
```