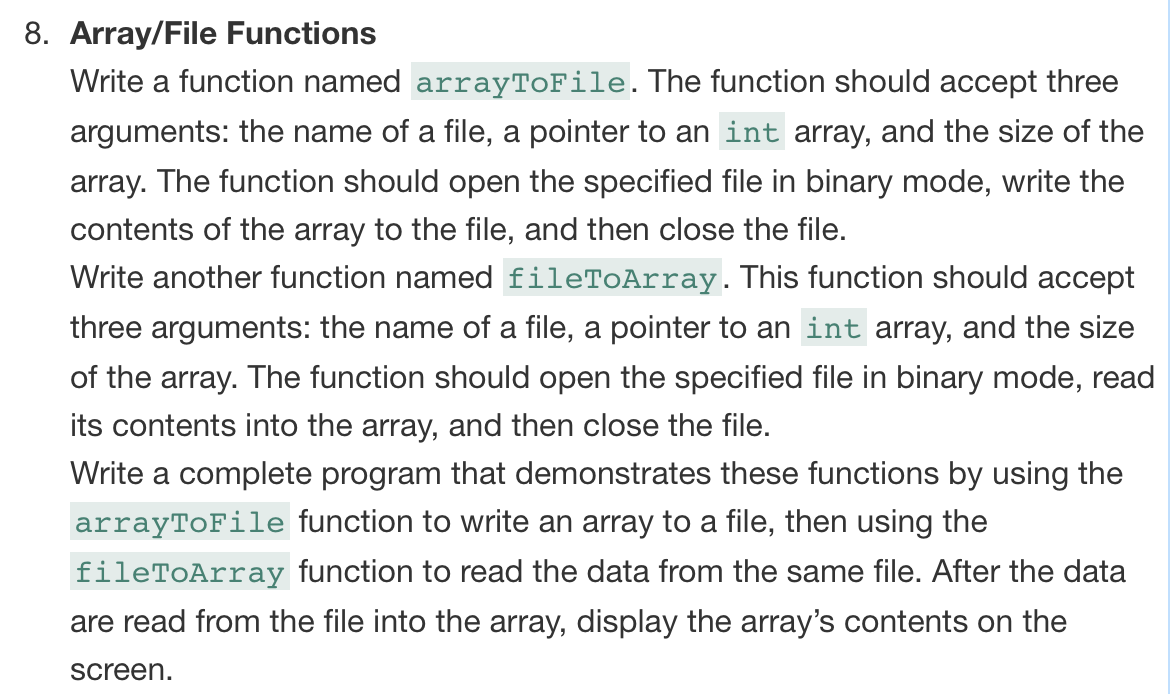
***COMSC165 Lab-4 Due: 7/15/18 11:59 PM***

Follow the lab format under the general folder in Canvas to complete this lab.



#ifndef FILEFUNCTION\_H

#define FILEFUNCTION\_H

#include <fstream>

class FileFunction

{

public:

char\* arrayPointer; //used to initialize file

char\* outputFromFile; //used to hold contents from file

const int arraySize = 5;

FileFunction(); //constructor use for initialization

// write array contents to file

void arrayToFile(std::fstream &, char \*, int);

// read file contents to array

void fileToArray(std::fstream &, char \*, int);

// display contents of the array

//displays array before being written to file

void displayArrayContents1(char \*, int);

//displays newly created array that gets contents from the file

void displayArrayContents2(char \*, int);

char\* getArray();

};

#endif // FILEFUNCTION\_H

#include "FileFunction.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

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

\* FileFunction constructor initializes two dynamic arrays \*

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

FileFunction::FileFunction() {

arrayPointer = new char[5]{'a','b','c','d','e'};

outputFromFile = new char[5]{'e','d','c','b','a'};

}

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

\* arrayToFile function accepts fstream object, array and int \*

\* as arguments then opens the file in binary mode and \*

\* writes the contents of the array to the file \*

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

void FileFunction::arrayToFile(fstream &fileObject, char \*arr, int arrSize){

// open the file for output in binary mode

fileObject.open("test.txt", ios::out | ios::binary);

// write the contents of the array to the file

cout << "writing array to file.\n";

fileObject.write(arrayPointer, arrSize);

// close the file

fileObject.close();

}

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

\* fileToArray function accepts fstream object, array, int \*

\* as argumentsthen opens the file in binary mode and \*

\* reads the contents of the file to the array \*

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

void FileFunction::fileToArray(fstream &fileObject, char \*arr, int arrSize){

// open the file for input in binary mode

fileObject.open("test.txt", ios::in | ios::binary);

// read the contents of the file to the array

cout << "readying file to array/memory.\n";

fileObject.read(arr, arrSize);

// close the file

fileObject.close();

}

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

\* displayArrayContents accepts an array/int as an argument \*

\* then prints the contents of the array to the console \*

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

void FileFunction::displayArrayContents1(char \*arr, int arrSize){

if (arr == arrayPointer) cout << "TRUE" << endl;

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

cout << arrayPointer[i] << " ";

}

cout << endl;

}

void FileFunction::displayArrayContents2(char \*arr, int arrSize){

if (arr == outputFromFile) cout << "TRUE" << endl;

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

cout << outputFromFile[i] << " ";

}

cout << endl;

}

#include "FileFunction.h"

#include <iostream>

#include <fstream>

using namespace std;

int main(){

FileFunction ffo;

fstream fileObject;

ffo.arrayToFile(fileObject, ffo.arrayPointer, ffo.arraySize);

ffo.displayArrayContents1(ffo.arrayPointer, ffo.arraySize);

ffo.fileToArray(fileObject, ffo.outputFromFile, ffo.arraySize);

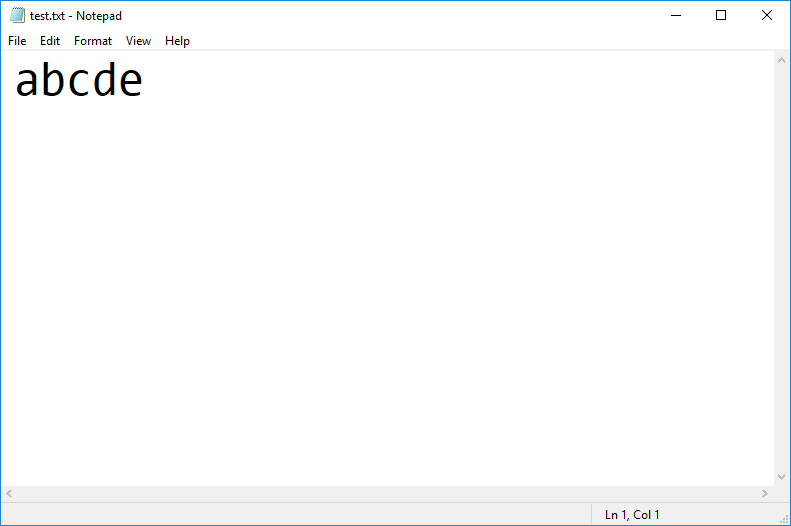
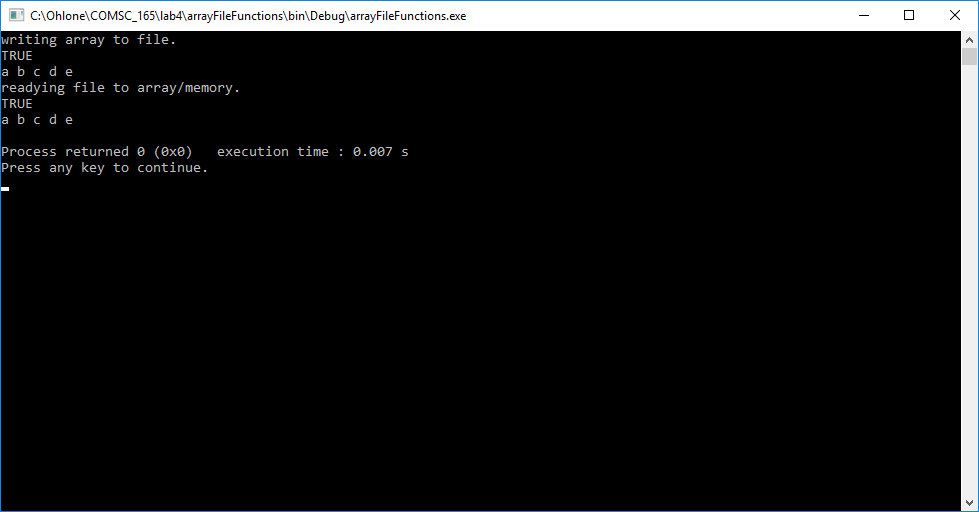
ffo.displayArrayContents2(ffo.outputFromFile, ffo.arraySize);

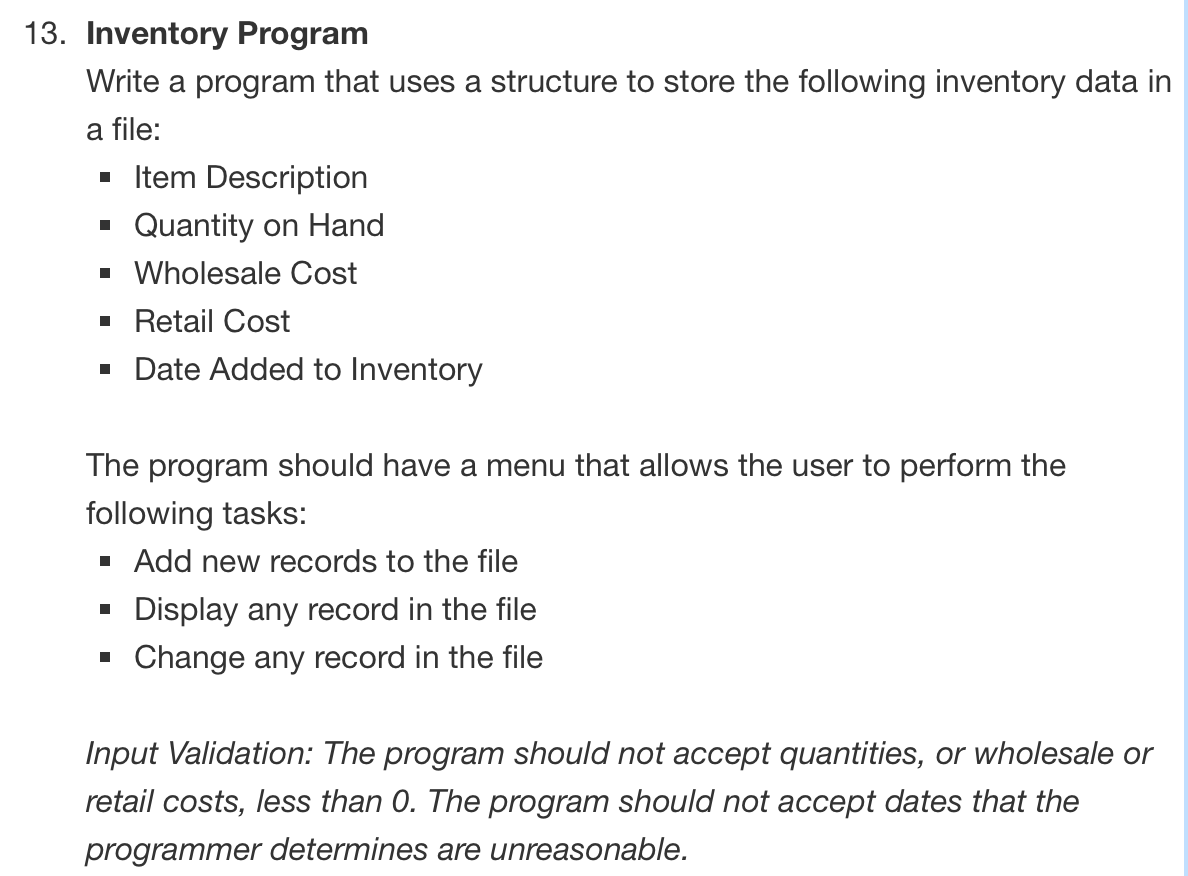
delete [] ffo.arrayPointer;

delete [] ffo.outputFromFile;

return 0;

}





#ifndef INVENTORYPROGRAM\_H

#define INVENTORYPROGRAM\_H

#include "InventoryItem.h"

#include <fstream>

class InventoryProgram

{

public:

int numItems;

// displays a menu for the user to interact with an item

int inventoryMenu();

// add new records to the file

void addNewRecords();

// display any record in the file

void displayRecords();

// change any record in the file

void changeRecords();

};

#endif // INVENTORYPROGRAM\_H

#ifndef INVENTORYITEM\_H

#define INVENTORYITEM\_H

#include <string>

const int DES = 80;

const int DATE = 12;

const int INTEGER = 21;

struct InventoryItem{

// item description

char description[DES];

// quantity on hand

char quantity[INTEGER];

// wholesale cost

char wholesaleCost[INTEGER];

// retail cost

char retailCost[INTEGER];

// date added to inventory

char dateAddedToInventory[DATE];

};

#endif // INVENTORYITEM\_H

#include "InventoryProgram.h"

#include "InventoryItem.h"

#include <iostream>

#include <string>

#include <fstream>

#include <cstdlib>

using namespace std;

// for the user to perform tasks

int InventoryProgram::inventoryMenu(){

int choice;

cout << "MENU" << endl;

cout << "1: Add new items to the inventory record." << endl;

cout << "2: Display inventory item." << endl;

cout << "3: Change any item in the inventory record." << endl;

cout << "4: Quit the program.\n" << endl;

cin >> choice;

while(choice < 1 || choice > 4){

cout << "You entered an invalid option. Please try again." << endl;

cin >> choice;

}

return choice;

}

// add new records to the file

void InventoryProgram::addNewRecords(){

// create fstream object

//open the file for output in binary mode

fstream fileObject("test.txt", ios::out | ios::binary);

if (!fileObject.is\_open()){

cout << "Error reading file!" << endl;

exit(EXIT\_FAILURE);

}else{

// create inventoryItem object

InventoryItem item;

char repeat;

int i = 0;

do{

// get user input for item information

cout << "Provide the item description: ";

cin.ignore();

cin.getline(item.description, DES);

cout << "Quantity: ";

cin >> item.quantity;

cout << "Wholesale cost: ";

cin >> item.wholesaleCost;

cout << "Retail cost: ";

cin >> item.retailCost;

cout << "Date added (mm/dd/yy): ";

cin.ignore();

cin.getline(item.dateAddedToInventory, DATE);

cout << endl;

// write contents of the item to the file

fileObject << endl;

fileObject << item.description << endl;

fileObject << item.quantity << endl;

fileObject << item.wholesaleCost << endl;

fileObject << item.retailCost << endl;

fileObject << item.dateAddedToInventory << endl;

cout << "Do you want to add another item? (y/n) ";

cin >> repeat;

}while(repeat == 'y');

cout << endl;

// close the file

fileObject.close();

}

}

// display any record in the file

void InventoryProgram::displayRecords(){

fstream fout("test.txt", ios::in | ios:: binary);

InventoryItem item;

// read contents of the item to the file

fout >> item.description;

fout >> item.quantity;

fout >> item.wholesaleCost;

fout >> item.retailCost;

fout >> item.dateAddedToInventory;

do{

// display item information

cout << "\nItem description:";

cout << item.description;

cout << "\nQuantity:";

cout << item.quantity;

cout << "\nWholesale cost:";

cout << item.wholesaleCost;

cout << "\nRetail cost:";

cout << item.retailCost;

cout << "\nDate added (mm/dd/yy):";

cout << item.dateAddedToInventory;

cout << endl;

// read contents of the item to the file

fout >> item.description;

fout >> item.quantity;

fout >> item.wholesaleCost;

fout >> item.retailCost;

fout >> item.dateAddedToInventory;

}while(!fout.eof());

// close the file

fout.close();

}

// change any record in the file

void InventoryProgram::changeRecords(){

fstream fin("test.txt", ios::out | ios:: binary);

InventoryItem item;

int itemNumber = 0;

// move to desired item and display

cout << "Enter the item # you'd like to edit: ";

cin >> itemNumber;

fin.seekg((itemNumber - 1)\*sizeof(item), ios::beg);

// get user input for item information

cout << "Enter new item info:" << endl;

cout << "Item description: ";

cin.ignore();

cin.getline(item.description, DES);

cout << "Quantity: ";

cin >> item.quantity;

cout << "Wholesale cost: ";

cin >> item.wholesaleCost;

cout << "Retail cost: ";

cin >> item.retailCost;

cout << "Date added (mm/dd/yy): ";

cin.ignore();

cin.getline(item.dateAddedToInventory, DATE);

cout << endl;

// write contents of the item to the file

fin << endl;

fin << item.description << endl;

fin << item.quantity << endl;

fin << item.wholesaleCost << endl;

fin << item.retailCost << endl;

fin << item.dateAddedToInventory << endl;

// close the file

fin.close();

}

#include "InventoryProgram.h"

#include "InventoryItem.h"

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

int main(){

InventoryProgram ipo;

int choice;

choice = ipo.inventoryMenu();

while(choice != 4){

switch(choice){

case 1:{

ipo.addNewRecords();

break;

}

case 2:{

ipo.displayRecords();

break;

}

case 3:{

ipo.changeRecords();

break;

}

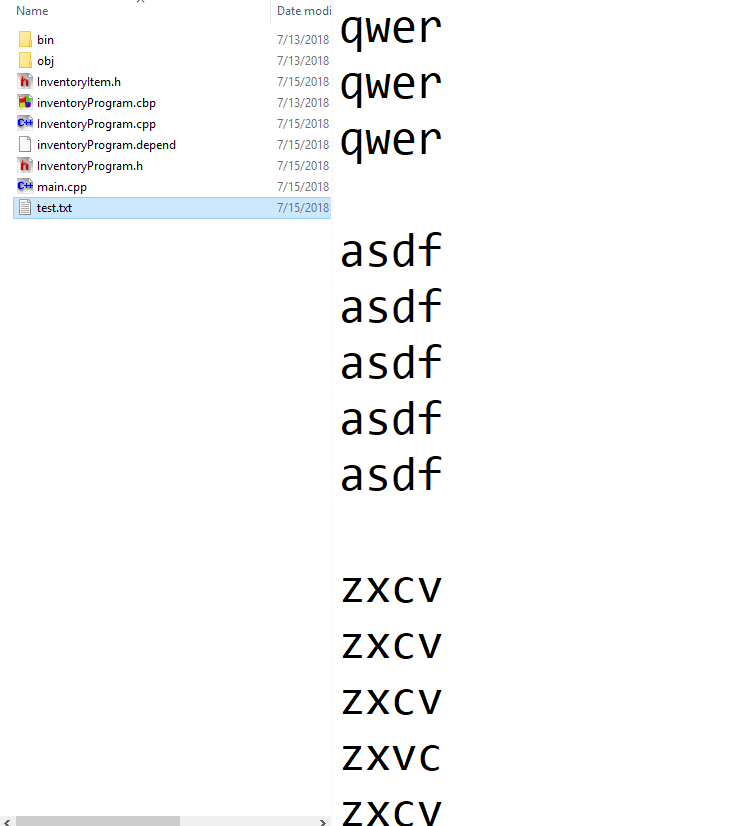
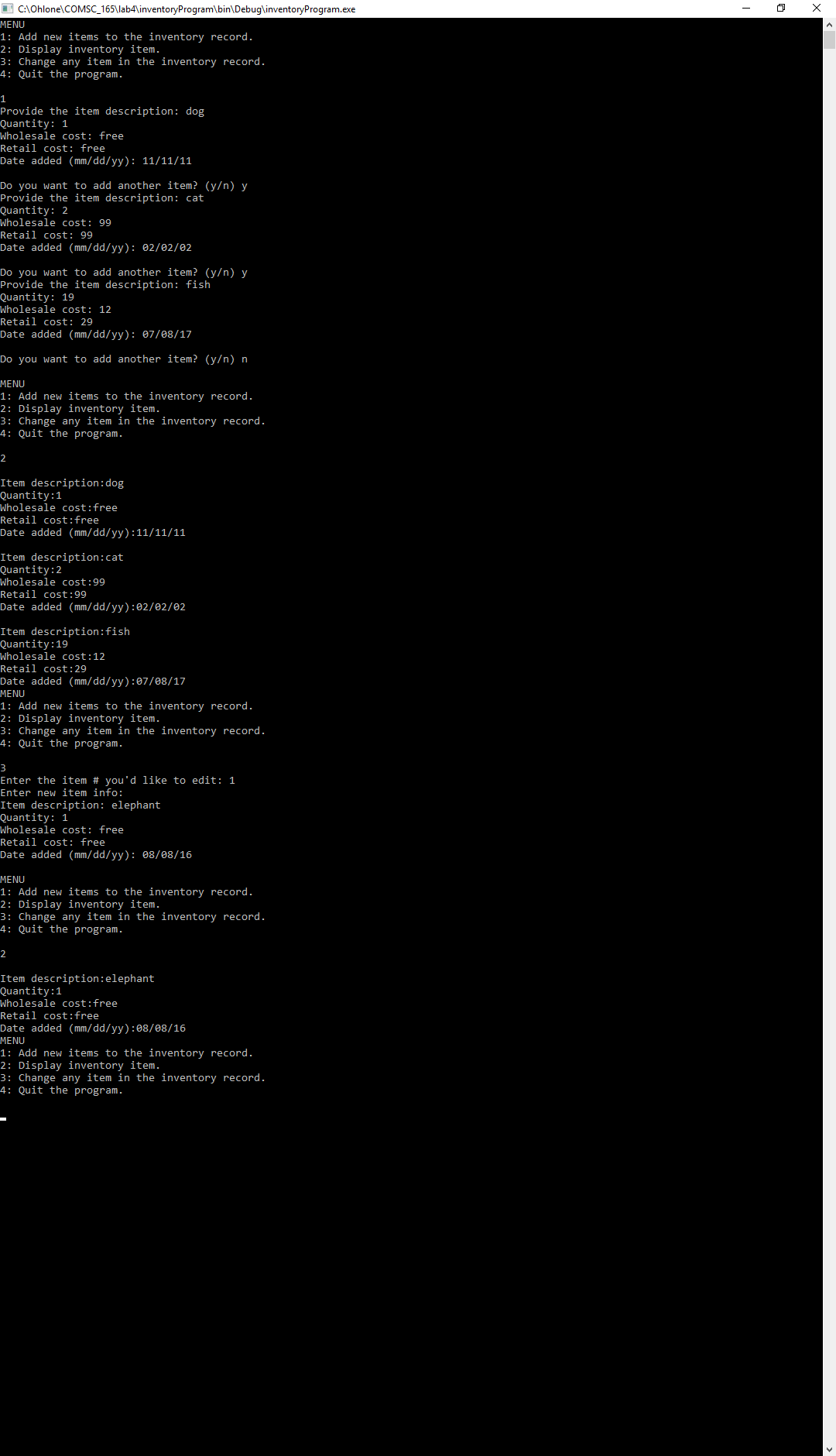
}

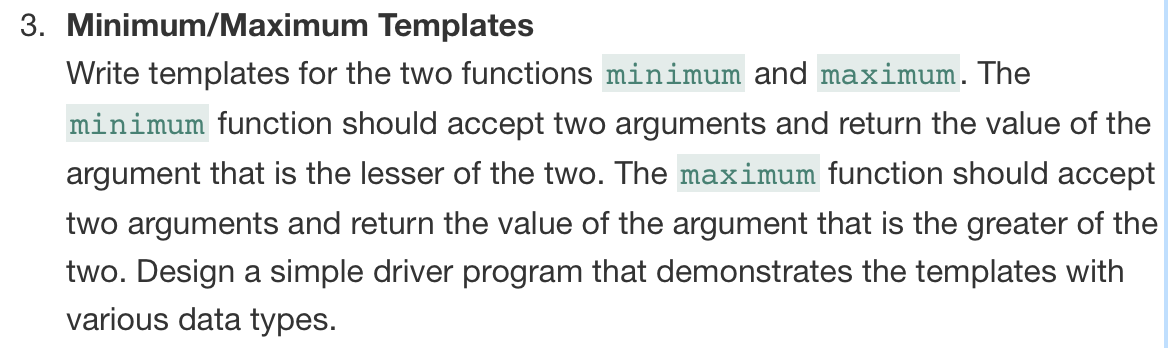
choice = ipo.inventoryMenu();

}

return 0;

}





#include <iostream>

using namespace std;

// declare function template

template <class TUNA>

// returns lesser of two arguments passed in

TUNA minimum(TUNA x, TUNA y){

if (x < y){

return x;

}else

return y;

}

//declare function template

template <class WATERMELON>

// returns greater of two arguments passed in

WATERMELON maximum(WATERMELON x, WATERMELON y){

if (x > y){

return x;

}else

return y;

}

int main(){

int x = 99;

int y = 100;

double a = 45.12;

double b = 50.98;

float pi = 3.1419238476;

float c = 99.9999999999;

cout << "minimum of 99 and 100 is: " << minimum(x, y) << endl;

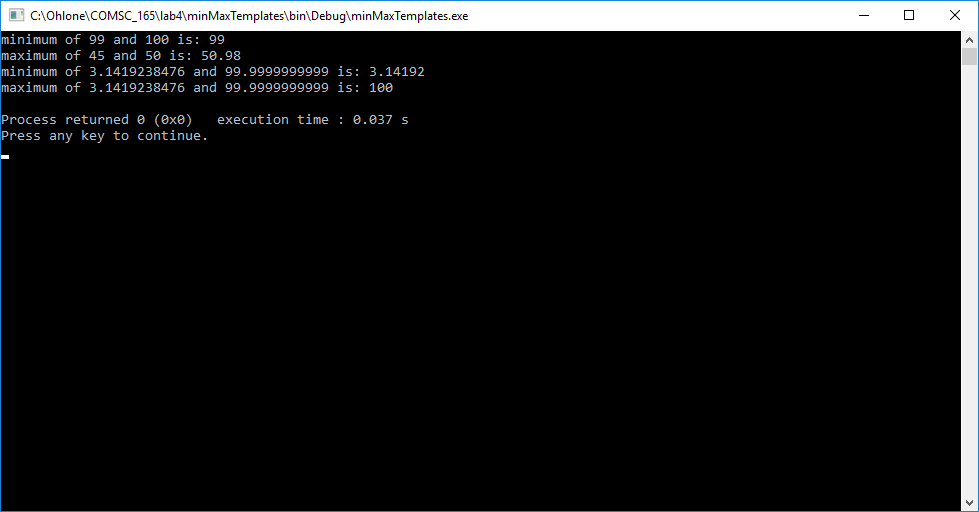
cout << "maximum of 45 and 50 is: " << maximum(a, b) << endl;

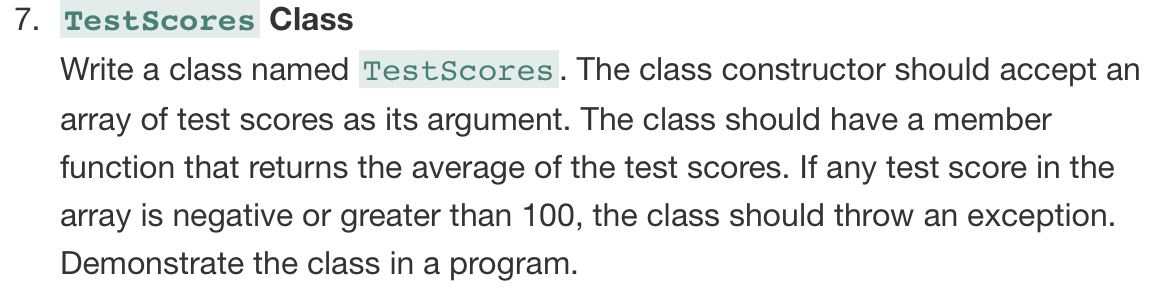
cout << "minimum of 3.1419238476 and 99.9999999999 is: " << minimum(pi, c) << endl;

cout << "maximum of 3.1419238476 and 99.9999999999 is: " << maximum(pi, c) << endl;

return 0;

}





#ifndef TESTSCORES\_H

#define TESTSCORES\_H

class TestScores{

public:

// constructor accepts array of test scores as parm

TestScores(double [], int);

// calculate average

double findAverage(double [], int);

// declare exception class

class InvalidScore{};

};

#endif // TESTSCORES\_H

#include "TestScores.h"

#include <iostream>

#include <string>

using namespace std;

// constructor accepts array of test scores as parm

TestScores::TestScores(double arr[], int arrSize){

try{

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

if (arr[i] <0 || arr[i] > 100){

throw InvalidScore();

}else{}

}

}

catch(InvalidScore){

cout << "\nInvalid test score";

}

}

double TestScores::findAverage(double arr[], int arrSize){

double averageScore = 0;

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

averageScore += arr[i];

}

averageScore /= arrSize;

return averageScore;

}

#include <iostream>

#include "TestScores.h"

using namespace std;

int main(){

const int ten = 10;

const int five = 5;

const int seven = 7;

double testOverOneHundredArray[ten] = {99, 100, 0, 88, 89, 90, 77, 92, 99, 101};

double testBelowZeroArray[five] = {99, 100, 0, 88, -1};

double testValidArray[seven] = {77, 77, 77, 77, 77, 77, 77};

cout << "The average for the test scores " << endl;

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

cout << testOverOneHundredArray[i] << ", ";

}

TestScores tso1(testOverOneHundredArray, ten);

cout << "is: " << tso1.findAverage(testOverOneHundredArray, ten) << endl;

cout << "The average for the test scores " << endl;

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

cout << testBelowZeroArray[i] << ", ";

}

TestScores tso2(testBelowZeroArray, five);

cout << "is: " << tso2.findAverage(testBelowZeroArray, five) << endl;

cout << "The average for the test scores " << endl;

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

cout << testValidArray[i] << ", ";

}

TestScores tso3(testValidArray, seven);

cout << "is: " << tso3.findAverage(testValidArray, seven) << endl;

return 0;

}

