**CSCI 1411 – Lab 13–Vectors of Objects**

**Goals:**

* Understand how to build and use Classes
* Understand how to use vectors of objects built with classes

**Development Environment:** (all students must use Clion)

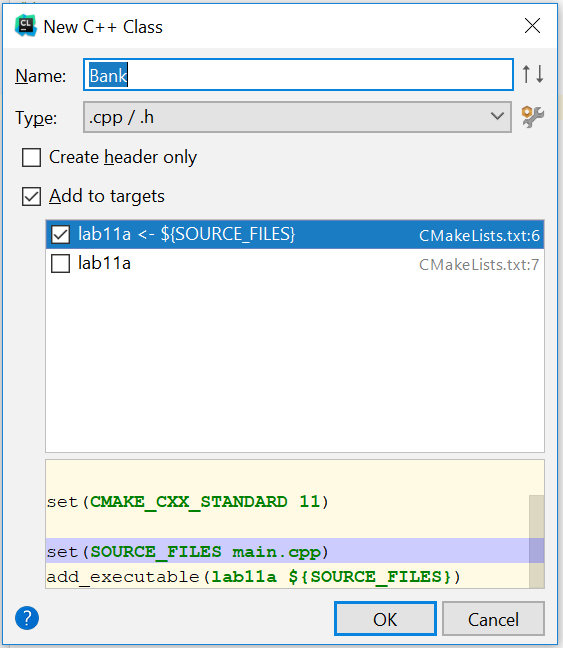
**Skills:** Classes, Default Constructor, Constructor, member functions, vectors of objects.

**Reading**: Chap 13

**Deliverables:** 1) This lab with 2 screen shots 2) lastnameFirstLab12.cpp, Scores.h, Scores.cpp

**Part I – Skills Practice (10 points)**

* Open a new project in CLion. Call the project Lab13a.
* For this project will make vectors of objects and use the built in member functions to add, and erase records.
* New->New C++ Class



* Add the name Bank. And accept the defaults Add to targets and Lab13a<-$(Source\_Files)
* This will create both a Bank.h file and a Bank.cpp file
* Double Click on the Bank.h file. Replace the entire text with this

**#ifndef BANK\_H  
#define BANK\_H  
  
#include <string>  
#include <iostream>  
using namespace std;  
class Bank  
{  
private:  
 string name;  
 int number;  
 float balance;  
public:  
 Bank();  
 Bank(string \_name, int \_number, float \_balance);  
 string getName(){return name;}  
 void setName(string \_name){name=\_name;}  
 int getNumber() {return number;}  
 void setNumber(int \_number){number = \_number;}  
 float getBalance(){return balance;}  
 void increaseBalance(float amount);  
  
};  
#endif**

* Now double click on the Bank.cpp. Replace the entire code with the code below:

**#include "Bank.h"  
Bank::Bank()  
{  
 name = "";  
 number = 0;  
 balance = 0;  
}  
  
Bank::Bank(string \_name, int \_number, float \_balance)  
{  
 name = \_name;  
 number = \_number;  
 balance = \_balance;  
}  
  
void Bank::increaseBalance(float amount)  
{  
 balance = balance + amount;  
}**

**New->New C++ class. Call it functions**

* **In functions.h**

**#ifndef FUNCTIONS\_H  
#define FUNCTIONS\_H  
#include "Bank.h"  
#include <vector>  
void bubbleSort(vector<Bank> & acctVector);  
ostream& operator << (ostream &out, Bank &tempBank);  
#endif**

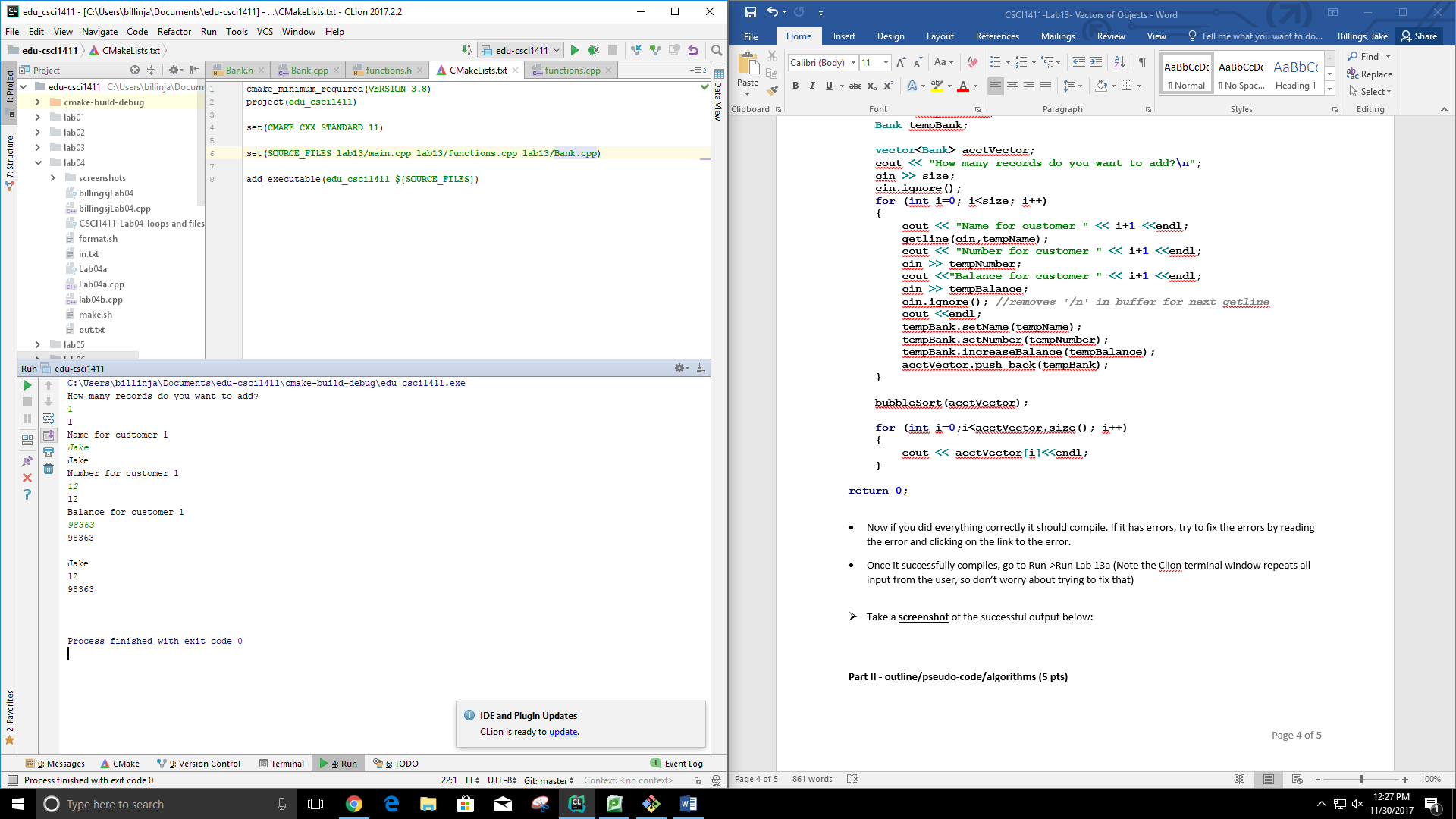
* **In functions.cpp**
* Now double click on the main.cpp. Replace the entire code with the code below:

**#include "functions.h"  
void bubbleSort(vector<Bank> & acctVector)  
{  
 *//Bubble Sort records by name* int maxElement;  
 int index;  
 Bank tempBank3;  
  
 for (maxElement = acctVector.size() - 1; maxElement > 0; maxElement--)  
 {  
 for (index = 0; index < maxElement; index++)  
 {  
 if (acctVector[index].getName() > acctVector[index + 1].getName())  
 {  
*//swap the entire record (name, number, balance)* tempBank3 = acctVector[index];  
 acctVector[index] = acctVector[index + 1];  
 acctVector[index + 1] = tempBank3;  
 }*//if* }*//for* }*//for*}  
  
ostream& operator << (ostream &out, Bank &tempBank)  
{  
 cout << tempBank.getName() << endl;  
 cout << tempBank.getNumber()<<endl;  
 cout << tempBank.getBalance()<<endl<<endl;  
 return out;  
}**

* In main.cpp

**#include <iostream>  
#include <vector>  
#include "Bank.h"  
#include "functions.h"  
using namespace std;  
int main()  
{  
 string tempName;  
 int tempNumber, size;  
 float tempBalance;  
 Bank tempBank;  
  
 vector<Bank> acctVector;  
 cout << "How many records do you want to add?\n";  
 cin >> size;  
 cin.ignore();  
 for (int i=0; i<size; i++)  
 {  
 cout << "Name for customer " << i+1 <<endl;  
 getline(cin,tempName);  
 cout << "Number for customer " << i+1 <<endl;  
 cin >> tempNumber;  
 cout <<"Balance for customer " << i+1 <<endl;  
 cin >> tempBalance;  
 cin.ignore(); *//removes '/n' in buffer for next getline* cout <<endl;  
 tempBank.setName(tempName);  
 tempBank.setNumber(tempNumber);  
 tempBank.increaseBalance(tempBalance);  
 acctVector.push\_back(tempBank);  
 }  
  
 bubbleSort(acctVector);  
  
 for (int i=0;i<acctVector.size(); i++)  
 {  
 cout << acctVector[i]<<endl;  
 }  
  
return 0;**

* Now if you did everything correctly it should compile. If it has errors, try to fix the errors by reading the error and clicking on the link to the error.
* Once it successfully compiles, go to Run->Run Lab 13a (Note the Clion terminal window repeats all input from the user, so don’t worry about trying to fix that)
* Take a **screenshot** of the successful output below:



**Part II - outline/pseudo-code/algorithms (5 pts)**

* Work with your lab partner to write an outline in comments and psuedocode to complete the following program. Use plain English for your outline.
* You should have one file broken into sections with comments. Each .h and .cpp should be in a separate section. Every program you write should have the following block at the top in comments. Make sure to fill in the Name, Class, Description and Lab Partner at the top of the file. Ensure your status is accurate.

/\* Name:  
Class: CSCI 1411-00X  
Description: [fill in description]

Lab Partner:  
Status: successfully compiled and run on csegrid [if it doesn’t run or meet all of the requirements, list the actual status!/\*

**billingsjLab13.cpp**

*/\*\*  
 \* Name: Jake Billings  
 \* Date: 11/30/2017  
 \* Class: CSCI 1411-001  
 \* Description: declare an array of 2 Scores, read in the 2 sets of 3 scores, Go through the array, call the .average()  
 \* member function for each set, and print the 3 values and the average for each set.  
 \* Status: compiles and runs on VS, Clion, and csegrid  
 \*/  
//---------Dependency Imports---------  
//Include cout  
//Include vectores  
//Include the Scores class  
//Include functions  
//Use the standard namespace  
//---------Declare Constants---------  
//-----------Main Function------------  
/\*\*  
 \* int main()  
 \*  
 \* The main entry point for the application.  
 \*  
 \* Returns 0.  
 \*/  
 //Declare a dynamically-sized vector  
 //Print a welcome message  
 //Iterate once for each space in the scores object array  
 //Print a message telling the user which object the scores are going into.  
 // Add one so that humans don't get confused by our zero indexing  
 //Read score one  
 //Read score two  
 //Read score three  
 //Create the Scores object  
 //Place the object in the array  
 //Print a message stating we are entering the averaging-phase of the progam  
 //Iterate through the array again to print the averages*

**Functions.cpp**

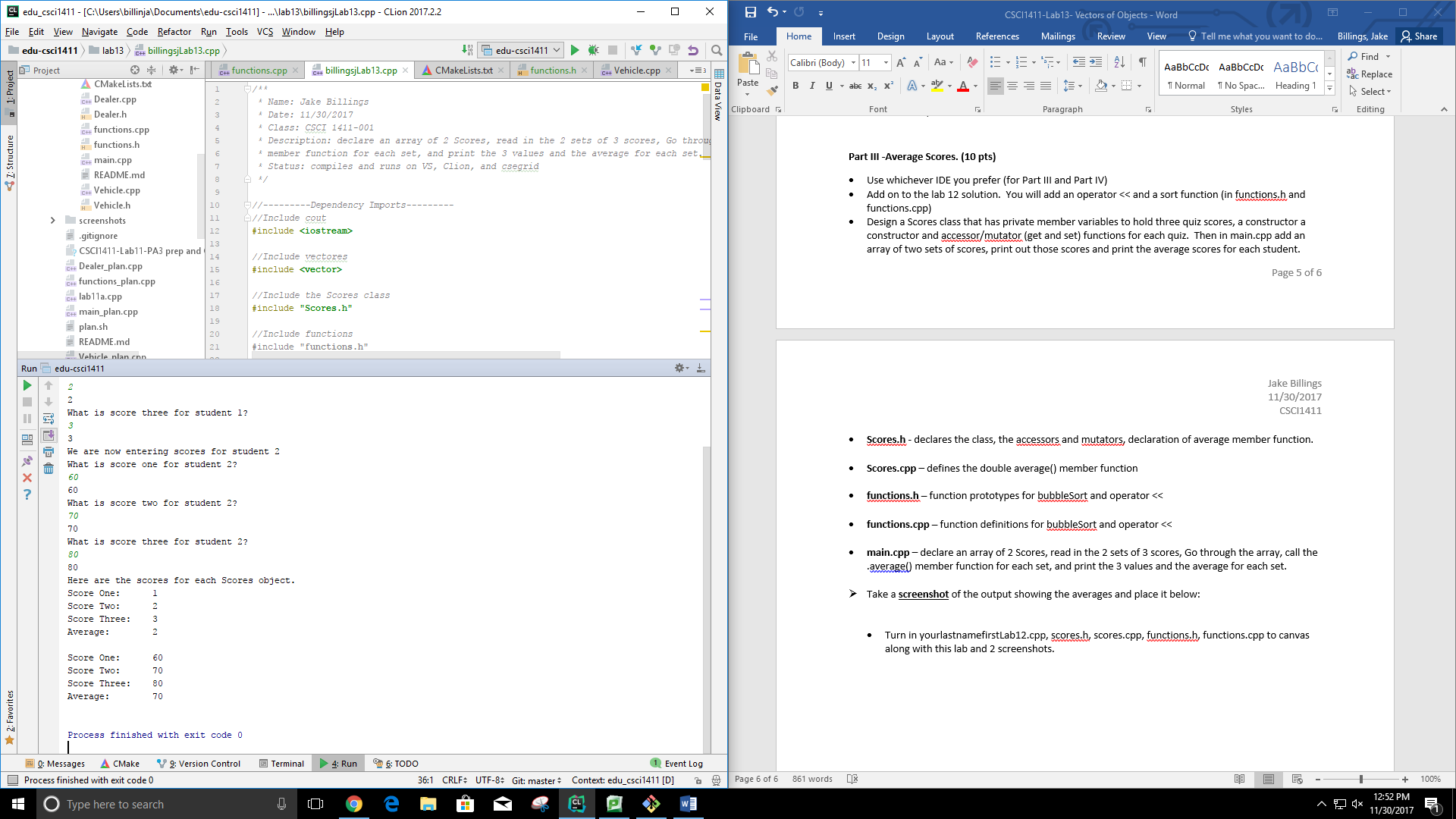
*/\*\*  
 \* Name: Jake Billings  
 \* Date: 11/30/2017  
 \* Class: CSCI 1411-001  
 \* Description: Function implementations for lab13  
 \* Status: compiles and runs on VS, Clion, and csegrid  
 \*/  
//Include the functions header, which includes dependencies  
//---------Implementations------------  
/\*\*  
 \* Implement << Operator for Scores object  
 \*  
 \* Prints the three scores and the average of them to an output stream.  
 \*  
 \* @param out An output stream to print to; usually cout  
 \* @param scores The scores object to print  
 \* @return Returns the out object  
 \*/  
 //Write to the output stream  
 //Return the output stream  
/\*\*  
 \* Implement bubbleSort() for vectors of Scores objects.  
 \*  
 \* Performs an in-place ascending bubbleSort on a vector of Scores objects using the  
 \* average score as a sort predicate.  
 \*  
 \* @param scoresVector  
 \*/  
 //Using bubble sort, arrange Scores from tallest to shortest in the inventory vector.*

**Scores.cpp**

*/\*\*  
 \* Name: Jake Billings  
 \* Date: 11/30/2017  
 \* Class: CSCI 1411-001  
 \* Description: Scores class implementation; the scores class holds three scores  
 \* Status: compiles and runs on VS, Clion, and csegrid  
 \*/  
//Implement constructors  
/\*\*  
 \* Implement an empty constructor  
 \*/  
/\*\*  
 \* Implement a full constructor that accepts all scores  
 \*  
 \* @param \_scoreOne The value to initialize scoreOne  
 \* @param \_scoreTwo The value to initialize scoreTwo  
 \* @param \_scoreThree The value to initialize scoreThree  
 \*/  
//Implement meaningful functions  
/\*\*  
 \* The getAverage() function returns the average of the three scores stored in this  
 \* Scores class object.  
 \*  
 \* @return a double representation of the average of the three scores  
 \*/  
//Implement Getters and Setters for each Score*

**Part III -Average Scores. (10 pts)**

* Use whichever IDE you prefer (for Part III and Part IV)
* Add on to the lab 12 solution. You will add an operator << and a sort function (in functions.h and functions.cpp)
* Design a Scores class that has private member variables to hold three quiz scores, a constructor a constructor and accessor/mutator (get and set) functions for each quiz. Then in main.cpp add an array of two sets of scores, print out those scores and print the average scores for each student.
* **Scores.h** - declares the class, the accessors and mutators, declaration of average member function.
* **Scores.cpp** – defines the double average() member function
* **functions.h –** function prototypes for bubbleSort and operator <<
* **functions.cpp –** function definitions for bubbleSort and operator <<
* **main.cpp** – declare an array of 2 Scores, read in the 2 sets of 3 scores, Go through the array, call the .average() member function for each set, and print the 3 values and the average for each set.



* Take a **screenshot** of the output showing the averages and place it below:
* Turn in yourlastnamefirstLab12.cpp, scores.h, scores.cpp, functions.h, functions.cpp to canvas along with this lab and 2 screenshots.