To: Professor Sargent and TA’s

From: Colin White

Date: May 29, 2018

Subject: Lab 4, Vectors

Enclosures.

a) Purpose of this lab

b) Your results for each of the sample data files provided on Learning Suite (included as an enclosure).

c) Discussion of the following:

1. Did your programs behave as expected for each of the sample data files, and why?

2. The advantages of using the vector class over static arrays.

d) Your source code.

e) Reports from the Final Check step.

The purpose of this lab: Write a program to read in two matrices from a file, perform various matrix operations, and print the result. Develop programming skills with arrays and matrices. Practice good documenting and programming style. Practice writing a lab memo.

The following are 8 screenshots of my results. They include each of the 4 sample files attempted to be added and then multiplied in chronological order. You will see that the correct ones were multiplied and the correct ones were added. Files that contained matrix that could not be added or multiplied ended with a message indicating so.

1. Did your programs behave as expected for each of the sample data files, and why?

My program did behave as expected. This is because it only added vectors when the number of rows and columns of matrix A equaled the number of rows and columns of matrix B. When this was not the case I got the proper indicating message. My program correctly also only multiplies matrix when then number of columns of matrix A equaled the number of rows of matrix B. When this was not the case I got the proper indicating message.

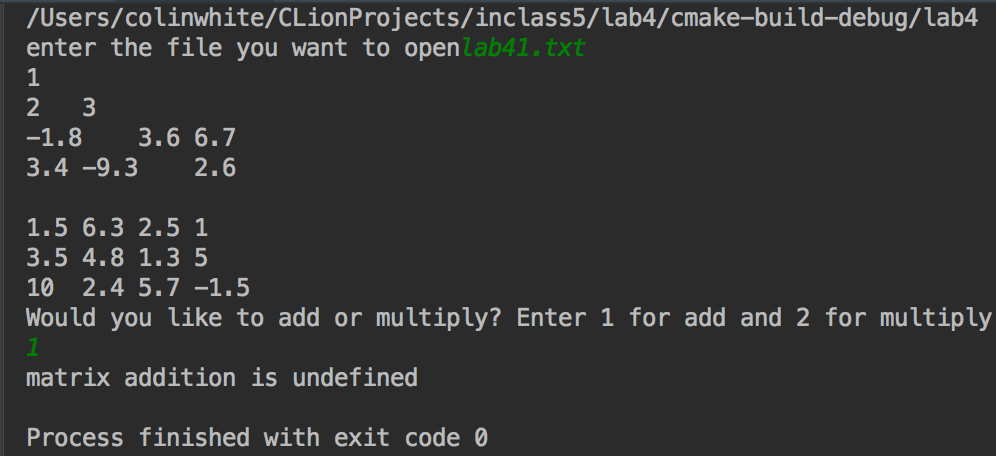
2. The advantages of using the vector class over static arrays.

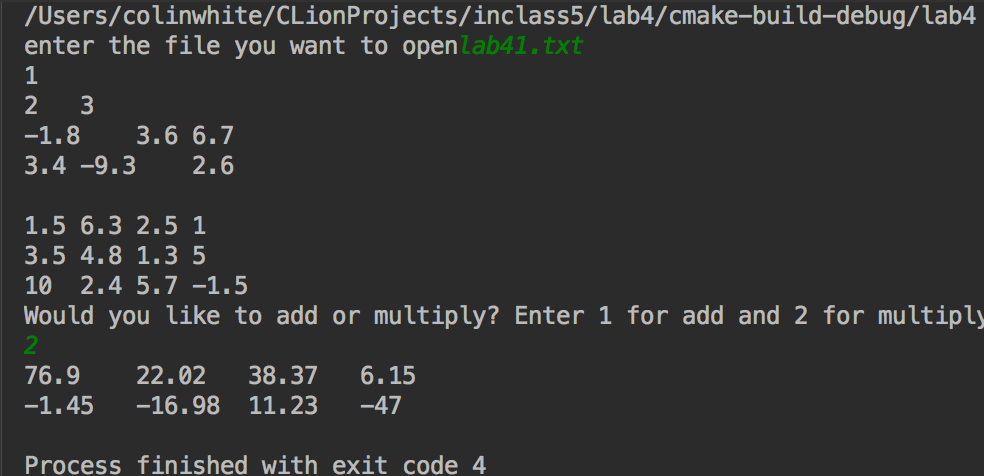
It is nice using vectors because they can dynamically allocate memory. They can expand as needed to fit the needed information from files that are being read in. These files are often of different lengths. Writing a program with vectors is nice so that one program can handle files of many different lengths.

e) Reports from the Final Check step.

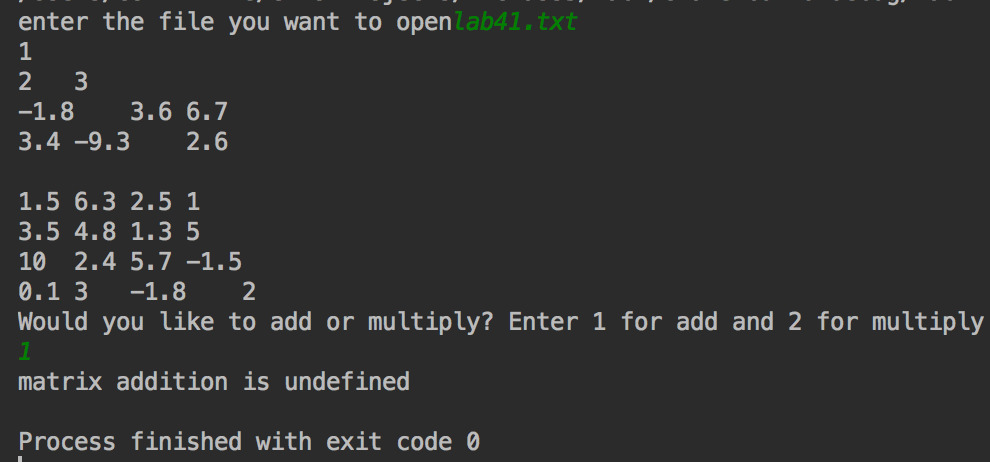
My final check confirmed that my c++ program was working correctly and as intended.

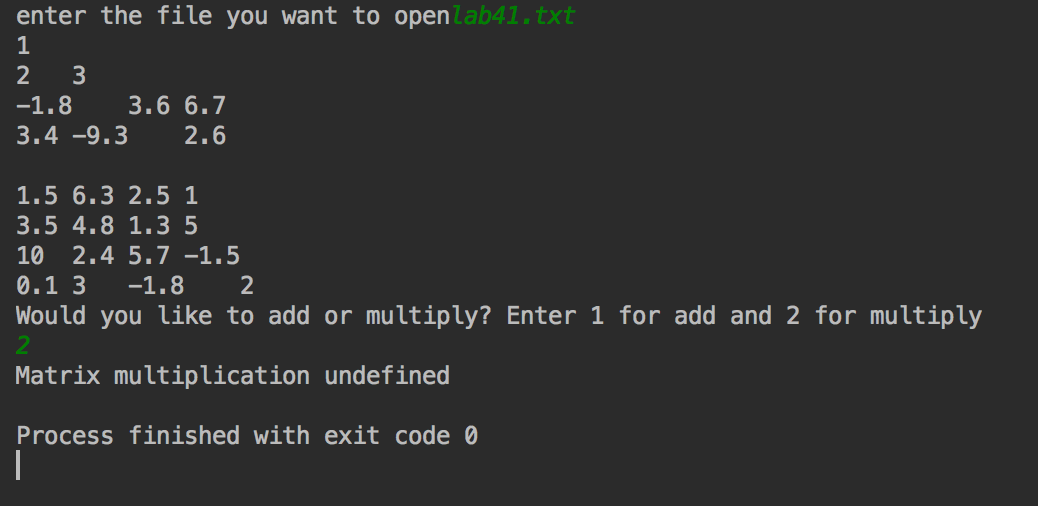
For Matrix 1 I got these results



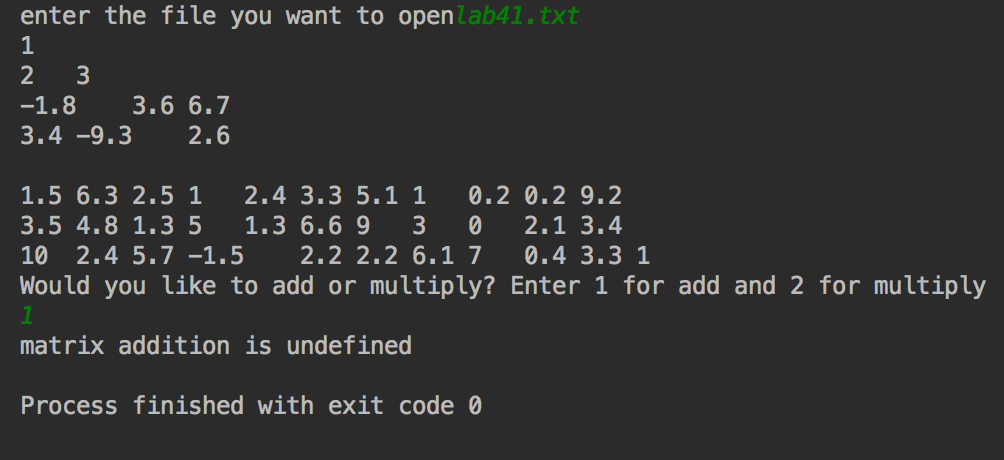


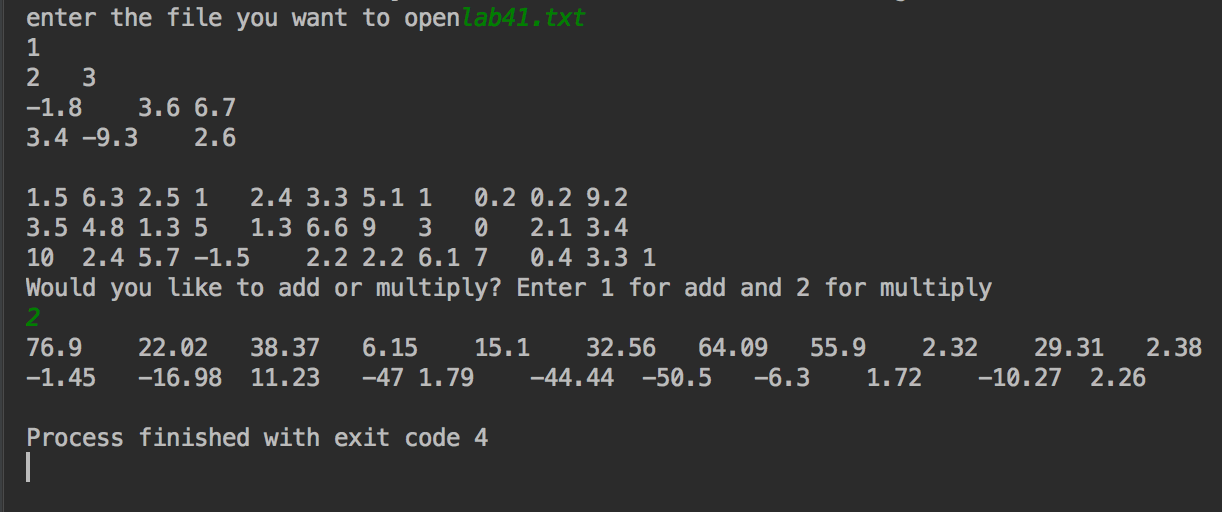
For Matrix 2 I got these results



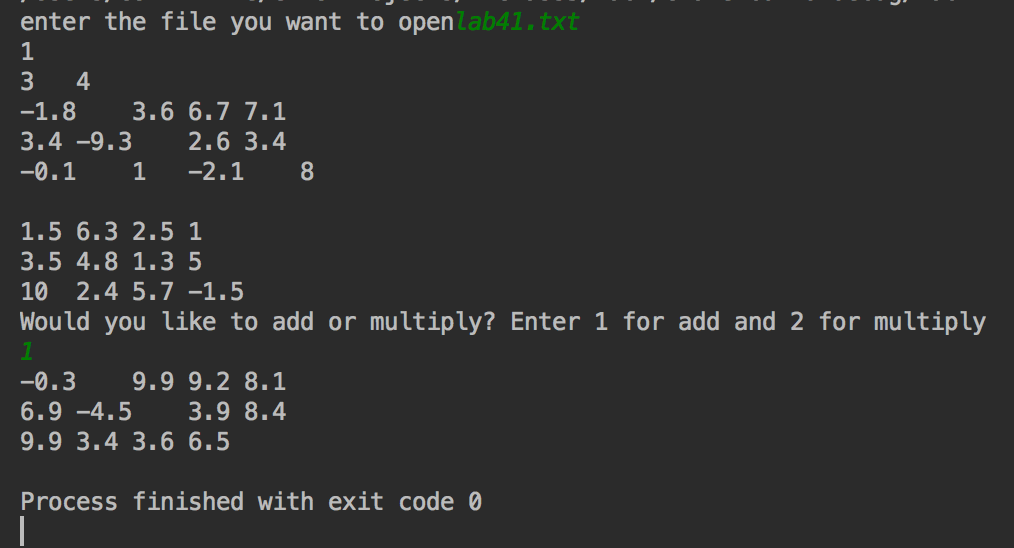


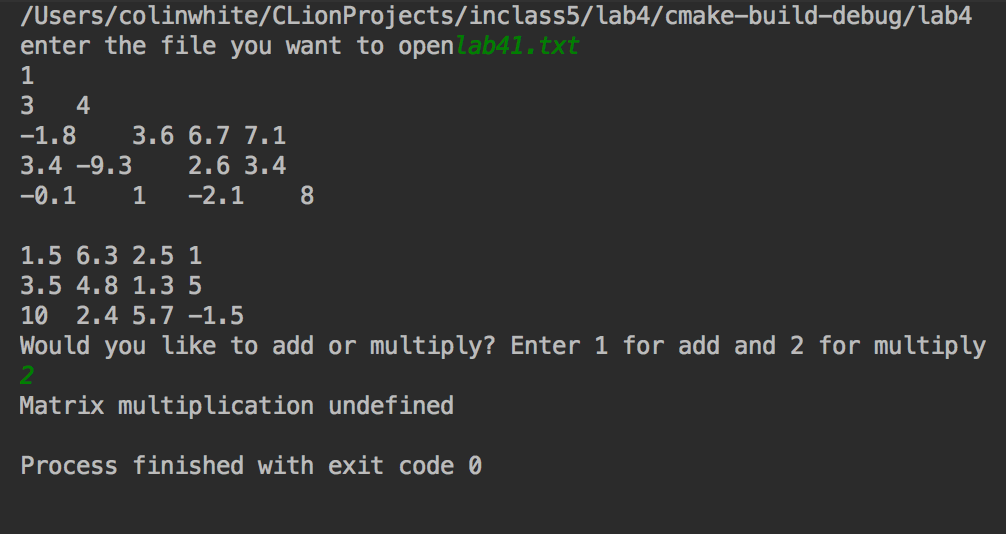
For Matrix 3 I got these results





For Matrix 4 I got these results





d) My source code.

/\*  
File name: Lab 4. Vectors  
Author: Colin White  
Date; 29 may 2018  
Description: This program asks a user what file they want to open. It reads an reprints the 2 matrix it sees in that file.  
 The program then askes if the user want to add or multiply the matrix. Depending on user input it calls the add  
 or mult function which add or multiply the matrix.  
  
 It will output the result, if the matrix are inconpatible, it will tell you.  
  
 The flaw in my program is that I could not get it to output to a file. It does the correct functions, but outputs it  
 to the interface, not a file.  
\*/  
#include <iostream>  
#include <fstream>  
#include <cmath>  
#include <string>  
#include <vector>  
**using namespace** std;  
  
vector<vector<**double**>> add(**double** rows1, **double** cols1, **double** rows2,**double** cols2, vector<vector<**double**>> mat1, vector<vector<**double**>> mat2); //prototyping the functions that will return addition of vectors  
vector<vector<**double**>> mult(**double** rows1, **double** cols1, **double** rows2,**double** cols2, vector<vector<**double**>> mat1, vector<vector<**double**>> mat2); //prototyping the functions that will return multiplicationof vectors  
  
ifstream fin; //Names fin as in input file stream  
ofstream fout; //names fout as the output file stream  
//the main function asks what file the user wants to open  
//depending on user input, the main function calls either the matrix addition or multiplication function  
**int** main()  
{  
 **double** rows1 = 0;//indicates rows of the first matrix  
 **double** cols1 = 0;//indicates columns of the first matrix  
 **double** rows2 = 0;//indicates rows of the secant matrix  
 **double** cols2 = 0;//indicates columns of the secant matrix  
 **int** input = 0; //the user inputs choice of operation  
 vector<vector <**double**>> mat1; //reads and stores matrix one  
 vector<vector <**double**>> mat2; //reads and stores matrix two  
  
  
 string filename; //declaring filename as variable type string. It will store what file the user wants to open  
 cout << "enter the file you want to open"; //ask user for the file they want to open  
 fout << "enter the file you want to open"; //ask user for the file they want to open  
 cin >> filename; //the user inputs the file they want to open  
 fin.open(filename); //opens the file the user wants to use  
 cout<<fin.is\_open()<<endl; //opens the file the user wants to use  
 fout<<fin.is\_open()<<endl; //opens the file the user wants to use  
 fin >> rows1 >> cols1; //program reads how many rows and columns the first matrix has  
 cout << rows1 << "\t" << cols1<<endl; //checking to make sure the program has obtained the right number or rows and columbs  
 fout << rows1 << "\t" << cols1<<endl; //checking to make sure the program has obtained the right number or rows and columbs  
 **double** num=0;//declare nums as a double to use to do operation later.  
  
 //this for loop reads, and prints the first matrix from the input file  
 **for**(**int** i=1;i<=rows1;i++)  
 {  
 vector <**double**> vec;  
 **for**(**int** j=1;j<=cols1;j++)  
 {  
 fin>>num;  
 vec.push\_back(num);  
 cout << num <<"\t";  
 fout << num <<"\t";  
 }  
 mat1.push\_back(vec);  
 vec.clear();  
 cout <<endl;  
 fout <<endl;  
 }  
  
 fin>>rows2>>cols2;  
 cout<<endl;  
 fout<<endl;  
  
 //this for loop reads, and prints the second matrix from the input file  
 **for**(**int** i=1;i<=rows2;i++)  
 {  
 vector <**double**> vec;  
 **for**(**int** j=1;j<=cols2;j++)  
 {  
 fin>>num;  
 vec.push\_back(num);  
 cout << num <<"\t";  
 fout << num <<"\t";  
 }  
 mat2.push\_back(vec);  
 vec.clear();  
 cout <<endl;  
 fout <<endl;  
 }  
  
  
  
 //asks user what operation they want to perform  
 cout << "Would you like to add or multiply? Enter 1 for add and 2 for multiply" << endl;  
 fout << "Would you like to add or multiply? Enter 1 for add and 2 for multiply" << endl;  
 cin >> input;  
 //if they want to add, the add function is called  
 **if** (input ==1)  
 {  
 add(rows1,cols1,rows2,cols2,mat1,mat2);  
 }  
 //if they want to multiply the multiplication function is called  
 **else if** (input==2)  
 {  
 mult(rows1,cols1,rows2,cols2,mat1,mat2);  
 }  
 //if neither the addition or multiplicaiotn function is called, the user gets an error message.  
 **else** {  
 cout << "i did not understand, rerun program" << endl;  
 fout << "i did not understand, rerun program" << endl;  
 }  
  
 //after all the calculations are done, the in and out files are closed.  
 //fin.close(); //close file  
 //fout.close(); //close file  
 **return** 0;  
 //end of the program  
}  
  
  
//this is the function that is called in case of addition.  
vector<vector<**double**>> add(**double** rows1, **double** cols1, **double** rows2,**double** cols2, vector<vector<**double**>> mat1, vector<vector<**double**>> mat2)  
{  
 vector<vector<**double**>> sum;  
 **if** (rows1==rows2 && cols1==cols2)  
 {  
  
 **for**(**int** i=0;i<rows1;i++)  
 {  
 vector<**double**> vec;  
 **for**(**int** j=0;j<cols1;j++)  
 {  
 **double** plus = 0;  
 plus= mat1[i][j] + mat2[i][j];  
// cout<<"mat1 "<<mat1[i][j]<<endl;  
// cout<<"mat2 "<<mat2[i][j]<<endl;  
 vec.push\_back(plus);  
 cout << plus <<"\t";  
 fout << plus <<"\t";  
 }  
 sum.push\_back(vec);  
 vec.clear();  
 cout << endl;  
 fout << endl;  
 }  
  
  
 **return** sum;  
 }  
 //this is the message the user gets if the matrices they try to add are of the wrong dimension to do so.  
 **else** {  
 cout << "matrix addition is undefined"<<endl;  
 fout << "matrix addition is undefined"<<endl;  
 **return** mat1;  
 }  
}  
  
//this is the function that is called when the user want to do matrix multiplication  
vector<vector<**double**>> mult(**double** rows1, **double** cols1, **double** rows2,**double** cols2, vector<vector<**double**>> mat1, vector<vector<**double**>> mat2)  
{  
 **if**(cols1==rows2) {  
 vector<vector<**double**>> prod;  
 **for** (**int** i = 0; i < rows1; i++)  
 {  
 vector<**double**> vec;  
 **for** (**int** j = 0; j < cols2; j++)  
 {  
  
 **double** sum = 0;  
 **for** (**int** k = 0; k < cols1; k++)  
 {  
 sum = sum+mat1[i][k] \* mat2[k][j];  
 }  
 vec.push\_back(sum);  
 }  
 prod.push\_back(vec);  
 }  
  
 **for**(**int** i=0;i<rows1;i++)  
 {  
 **for**(**int** j=0;j<cols2;j++)  
 {  
 cout <<prod[i][j]<<"\t";  
 fout <<prod[i][j]<<"\t";  
 }  
 cout<<endl;  
 fout<<endl;  
 }  
 // return prod;  
 }  
  
 //this is the message the user gets if the matrices they try to multiply are of the wrong dimension to do so.  
 **else** {  
 cout << "Matrix multiplication undefined"<< endl;  
 fout << "Matrix multiplication undefined"<< endl;  
 **return** mat1;  
 }  
}