**ECE 7650 ASSIGNMENT REPORT**

**(C++ and Object Oriented Computing for Engineers)**

**Author:** Jamiu Babatunde Mojolagbe

**Department:** Electrical and Computer Engineering

**Student ID:** #7804719

**Email:** mojolagm@myumanitoba.ca

**Course:** ECE 7650

**Sub. Date:** February 12, 2018

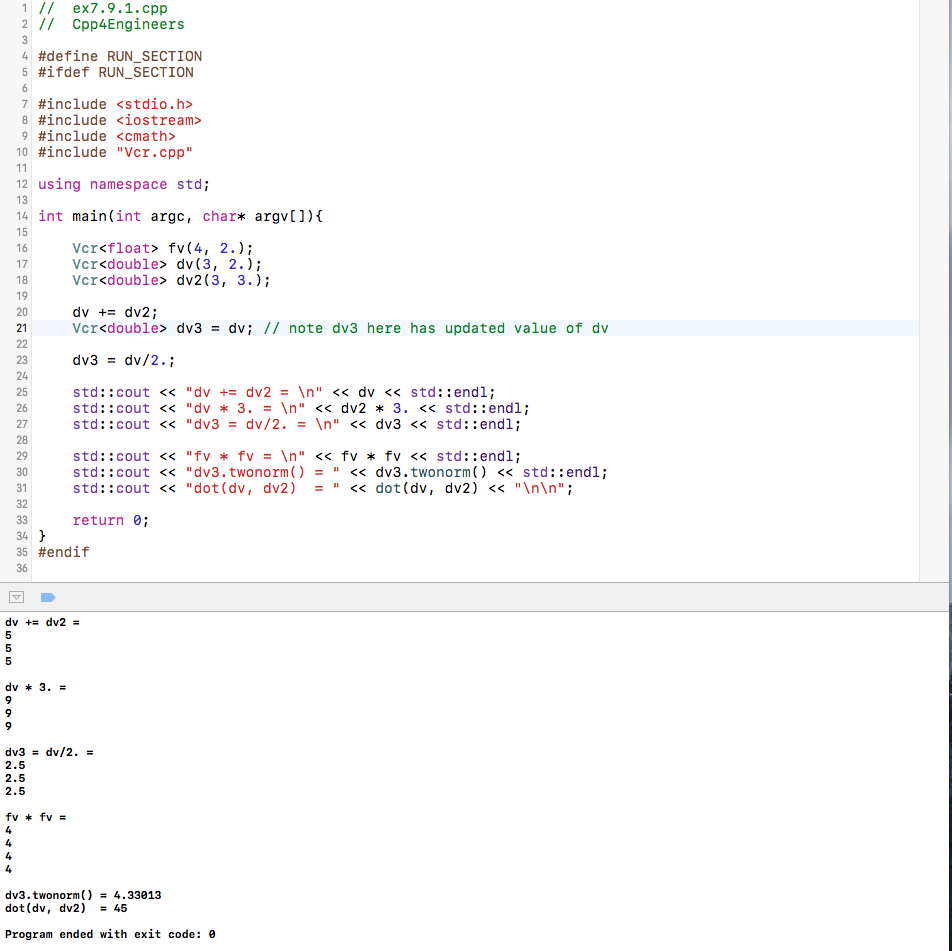
**QUESTION (7)**

All codes as regards to this question 7 are contained in folder named “**Chapter7**”.

**Question 7.9.1**

The class template for vectors as illustrated in §7.1 with additional operations such as vector addition and vector multiplication are contained in the files named “**Vcr.h**” and “**Vcr.cpp**”. While the “Vcr.h” is the header file containing the class definition with the class data and function signatures; “Vcr.cpp” contains the implementation of the class template itself.

The driver program for this vector class implementation is contained in the file named “**ex7.9.1.cpp**”. The content of the driver program and the results obtained are shown below:

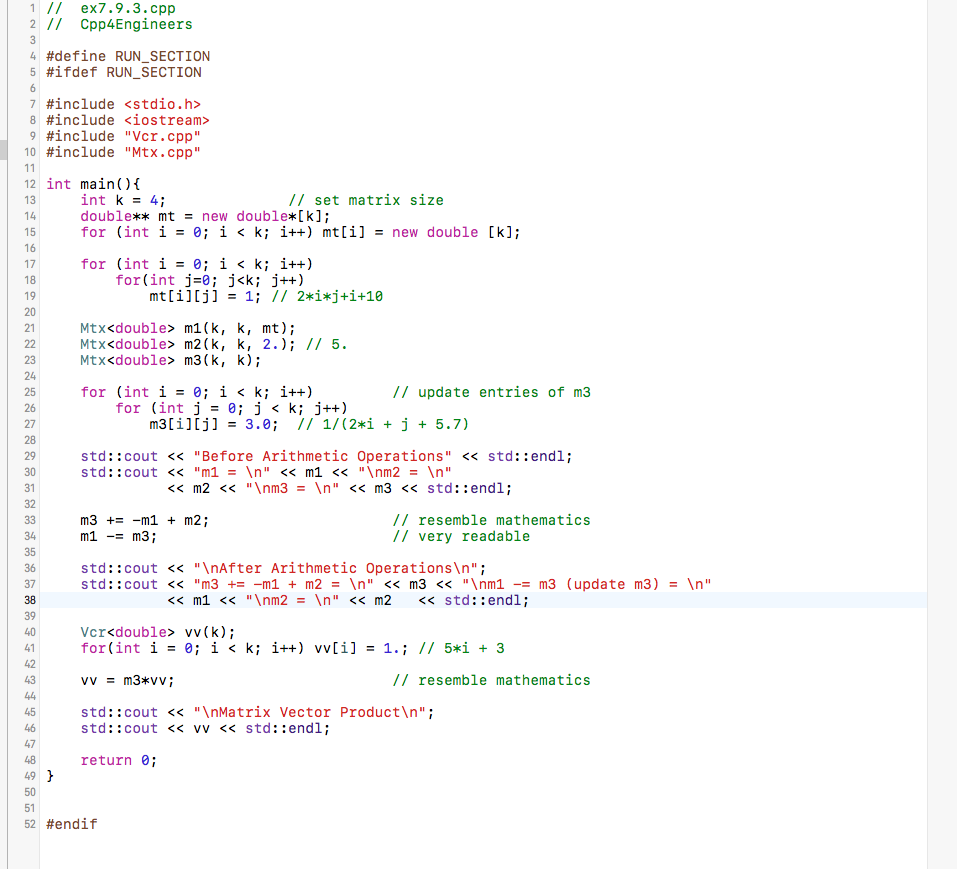


**Question 7.9.3**

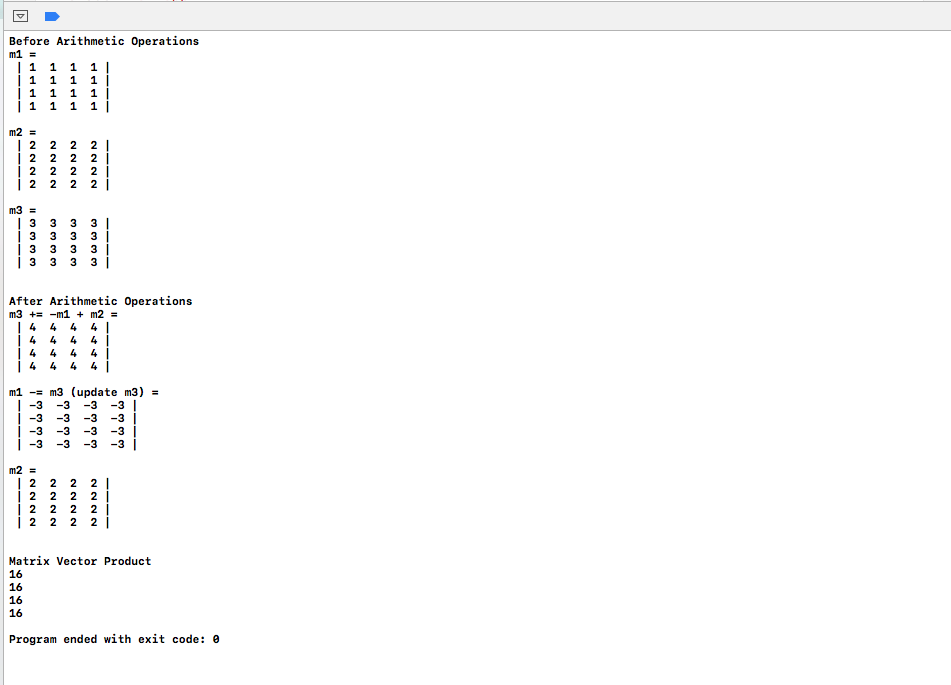
The matrix class defined in §6.3 was implemented as a template class. The **Mtx** class template header file and definition file are named “**Mtx.h**” and “**Mtx.cpp**” respectively. However, the template implementation in these files also contain template specialization for complex case involving **Question 7.9.15**.

The driver code – named “**ex7.9.3.cpp**” - and the output window are shown in the figures below:

Driver Program:



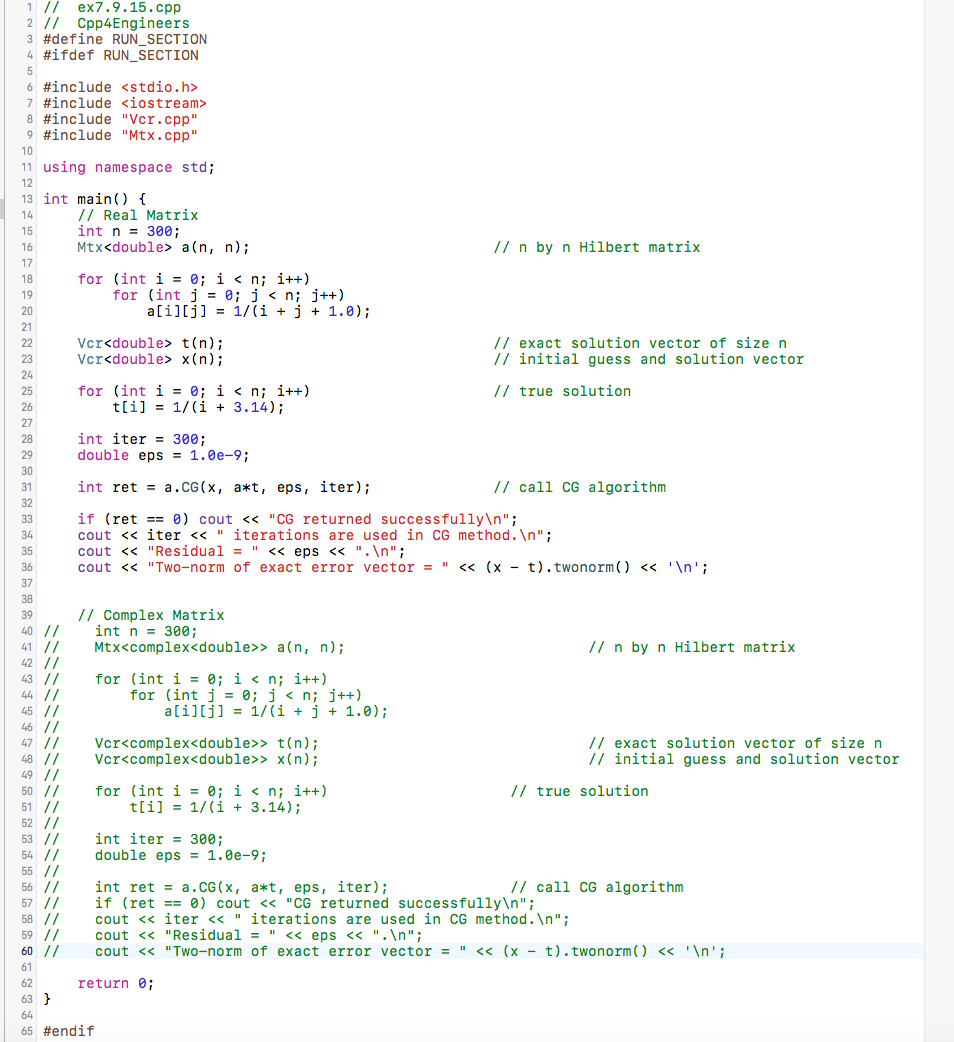
Output Window:



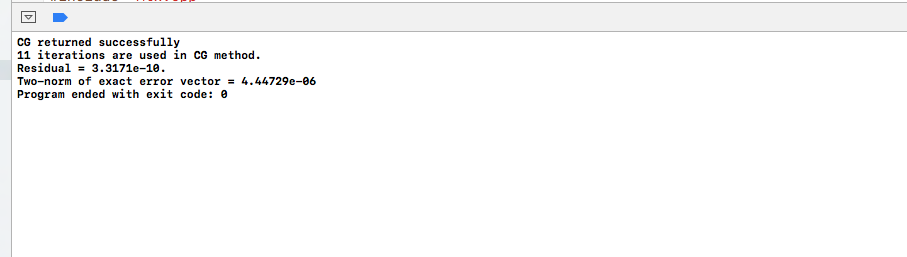
**Question 7.9.15**

The implementation of this question is partially contained in the “**Vcr.cpp**” and “**Mtx.cpp**”. In both of these files the complex specialization of vector operations and complex specialization of matrix operations are implemented for simplicity – so as to avoid duplicate codes. Again, Conjugate Gradient, CG, is implemented in the “**Mtx**” template class for both the real and complex precisions.

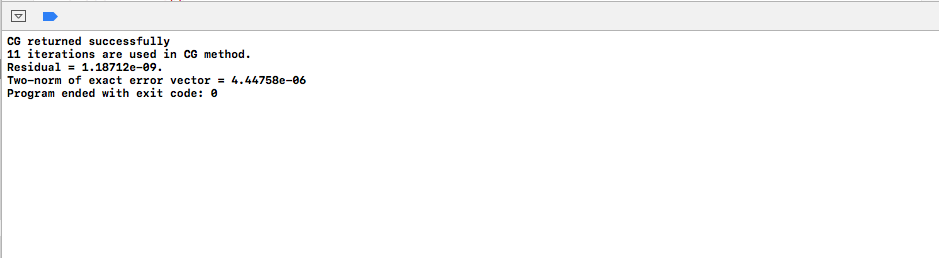
Therefore, the file named “**ex7.9.15.cpp**” is the driver program for this question. The driver program is coded to test for both real and complex cases with known solution as depicted in the C++ Textbook on page 228 (Section 6.6). Below is the driver’s screenshot:



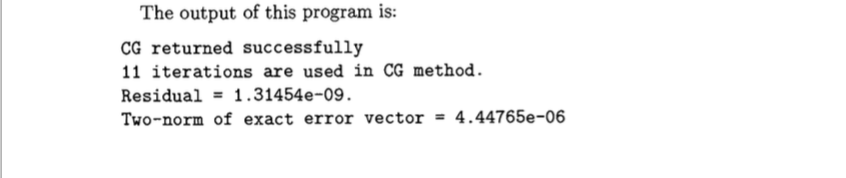
Output Screen for Real Matrix:



Output Screen for Complex Matrix with zero (0) imaginary part for the same problem (as the above):



The result obtained by the author of the textbook as contained in page 228 is:

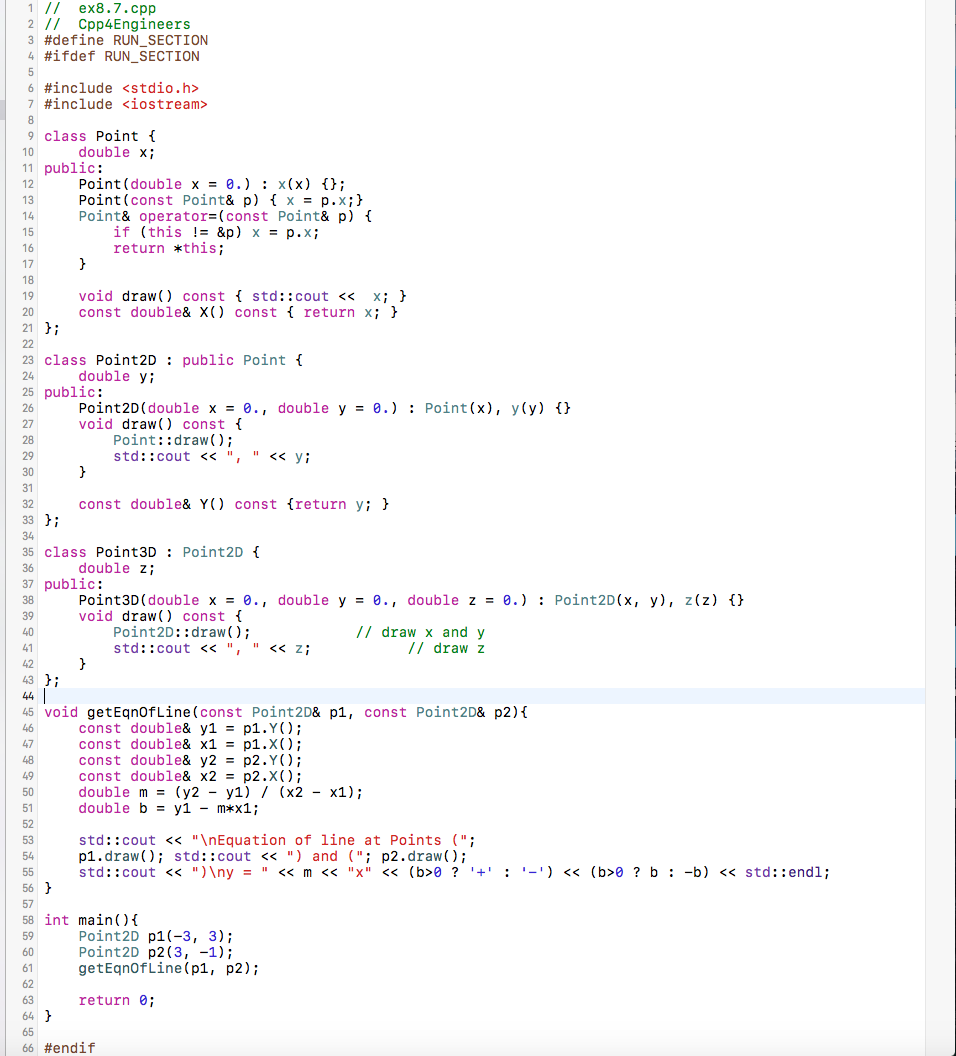


**QUESTION (8)**

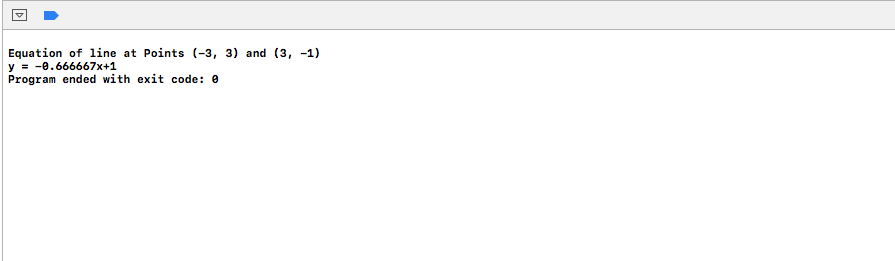
All codes as regards to this question 8 are contained in folder named “**Chapter8**”.

**Question 8.7.1**

The class hierarchy for points 1D, 2D and 3D are implemented in the file named “**ex8.7.1.cpp**”. As stated in the question, the function for finding equation of line passing through two (2) points is also implemented in the same file. This additional function solves the equation of line and prints out the resulting equation on the screen. The implementations and driver codes are shown below:



Output Window:



**Question 8.7.3**

A virtual destructor is a virtual function defined in the base class of a derived class. A virtual destructor is used such that proper cleanup in the allocated memory is achieved. Such use is required when an object or a class of the derived class is manipulated through the base class's pointer and both the superclass and sub-class have dynamic memory allocations; This implies that there is run-time polymorphism. However, there is no need of virtual destructor if and only if the derived class is not manipulated through the base class's pointer; In this case memory allocation will be properly deallocated.

Concisely, a class with run-time polymorphism (virtual functions) needs to have virtual destructor.

Example: Consider a derived class B having a base class A such that memory is allocated in both A and B. Expression like:

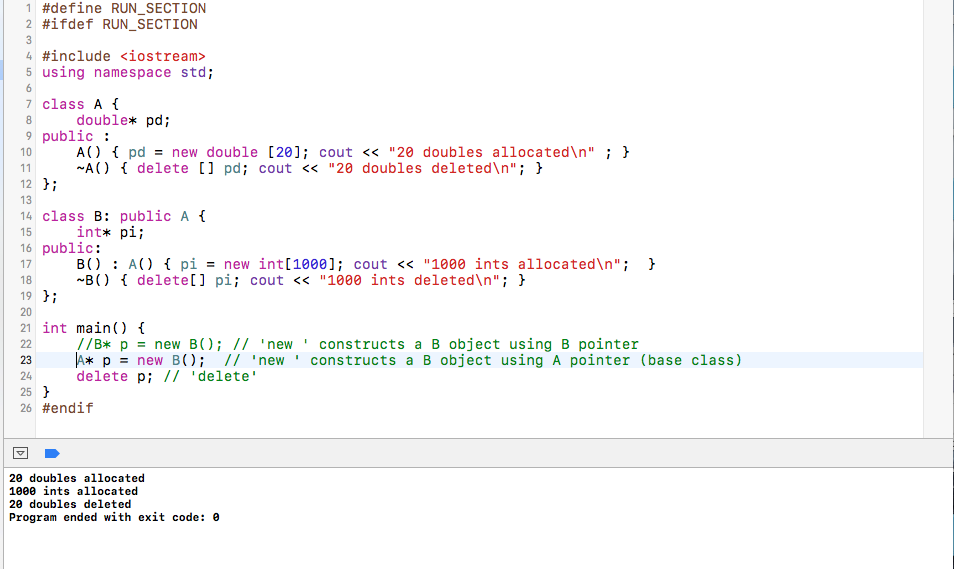
A\* obj = new B( ); // manipulation of derived class through the base class

delete obj; // proper cleanups are not achieved here if the base destructor is not virtual

To achieve the desired memory deallocation, the destructor in base class A should be made virtual and as such proper cleanups can be achieved in B.

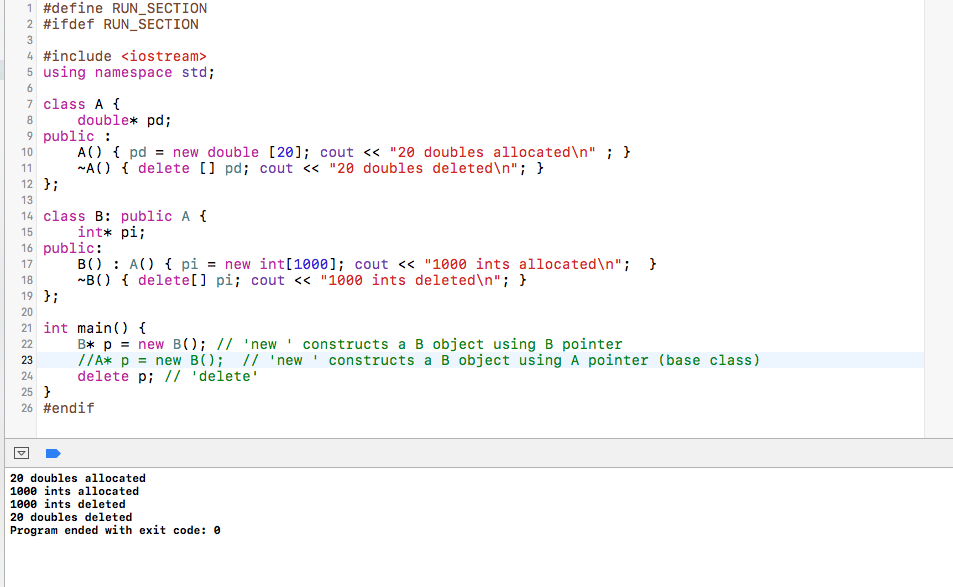
To demonstrate the above assertions, a short driver as well as classes programs was written as contained in the file named “**ex8.7.3.cpp**”.

**CASE 1:** When an object of the derived class is manipulated by the base class’s pointer but the destructor of the base class is not a virtual function.



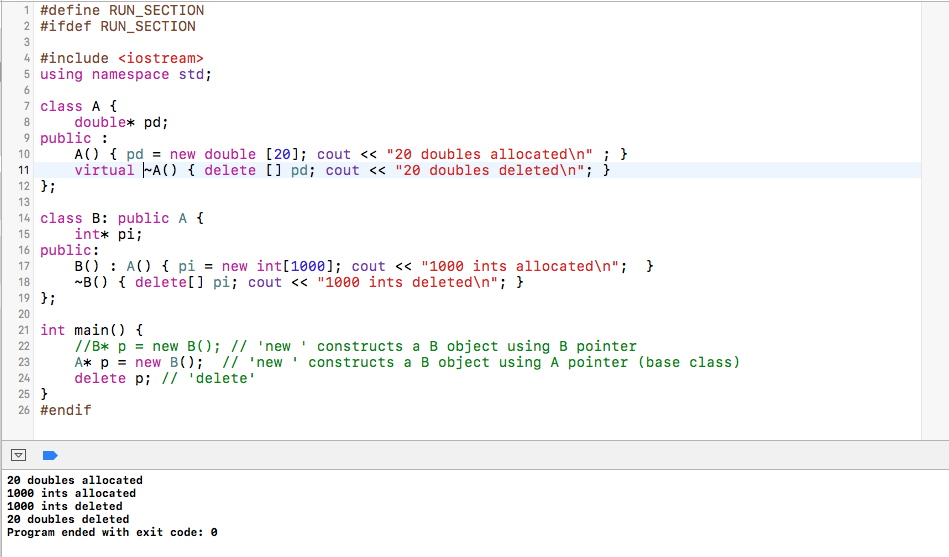
**Note:** Not all allocated memory spaces are freed - Improper cleanup.

**CASE 2:** When an object of the derived class is manipulated by the derived class’s pointer but the destructor of the base class is not a virtual function.



**Note:** Here proper cleanups was achieved as all allocated memory spaces got freed.

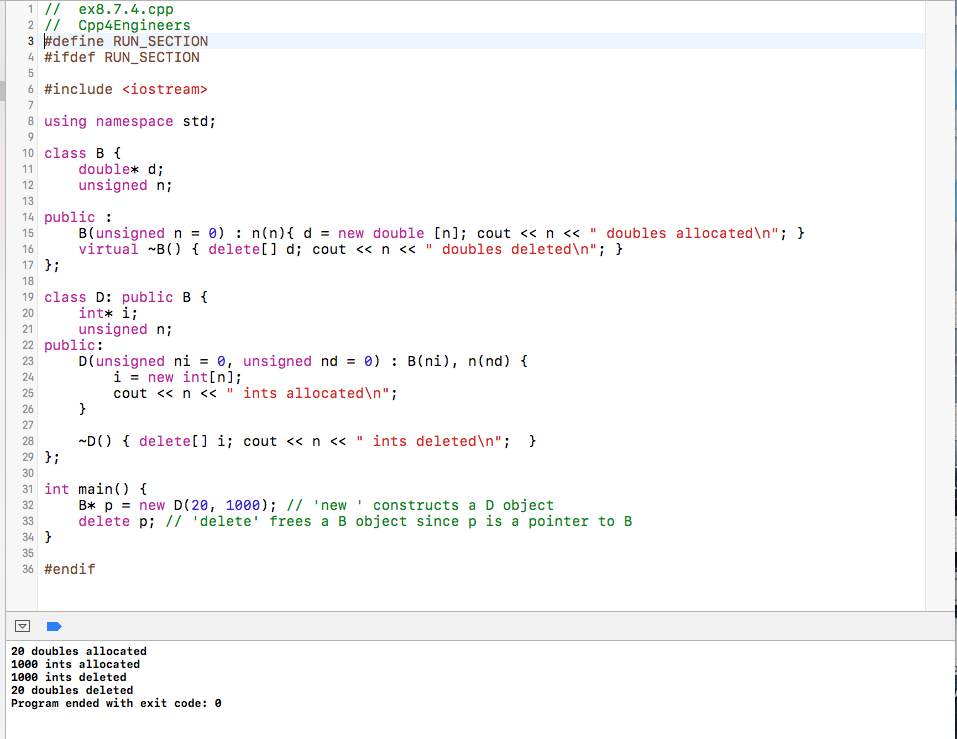
**CASE 3:** When an object of the derived class is manipulated by the base class’s pointer and the destructor of the base class is a virtual function.



**Note:** Here proper cleanups was achieved as all allocated memory spaces got freed.

**Question 8.7.4**

“**ex8.7.4.cpp**” is the implementation file of this question. Below is the code and the output screen.

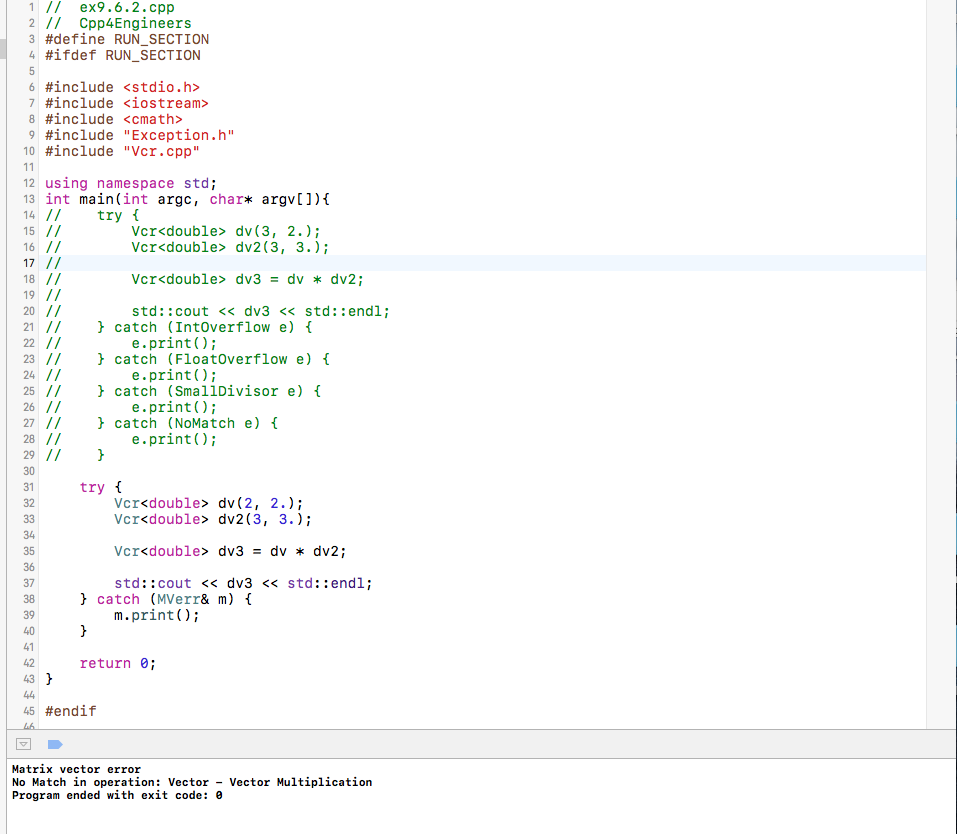


**QUESTIONS (9 & 10)**

All codes as regards to these questions are contained in folder named “**Chapter9\_10**”.

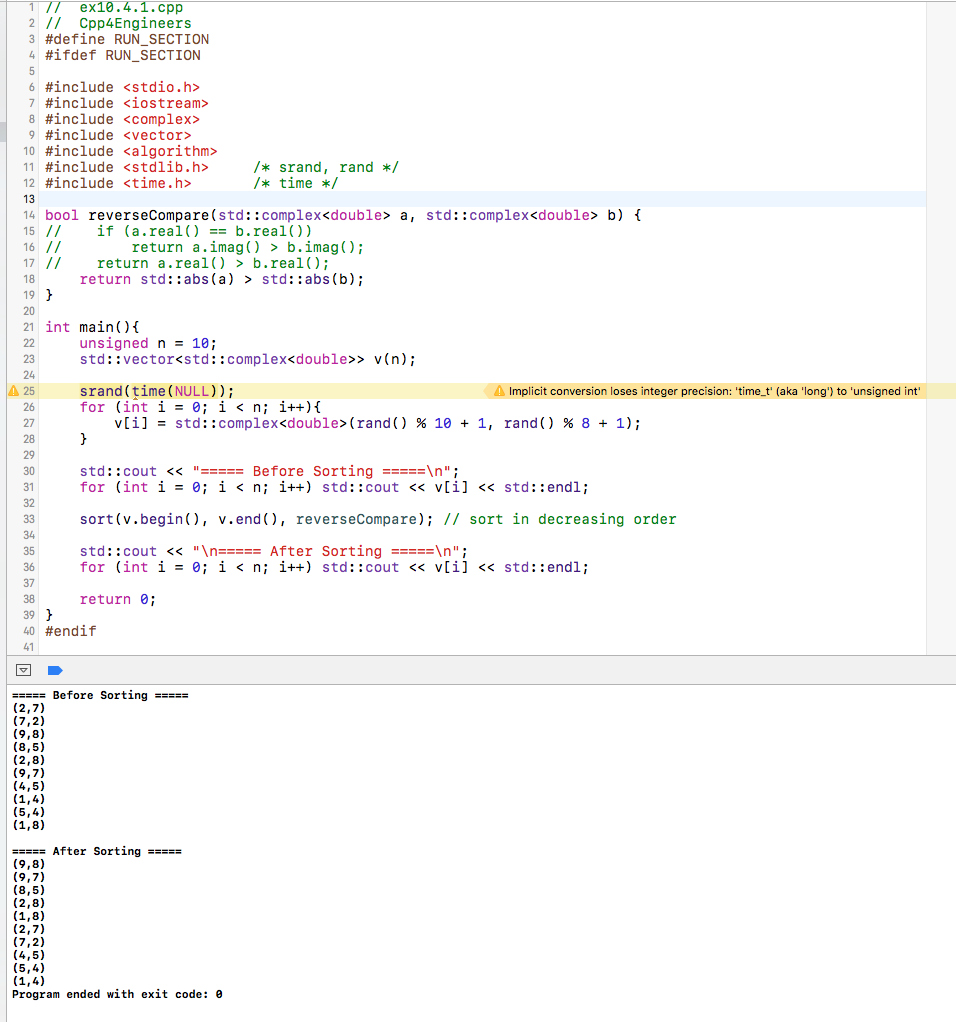
**Question 9.6.2**

Vector template for throwing an exception was implemented in “**Vcr.h**” and “**Vcr.cpp**”. While the class hierarchical exception handler is implemented in the file named “**Exception.h**”. The driver program for this question is named “**ex9.6.2.cpp**”. Here, there is a naïve implementation (commented part) and efficient implementation using catch with base class’s object reference.



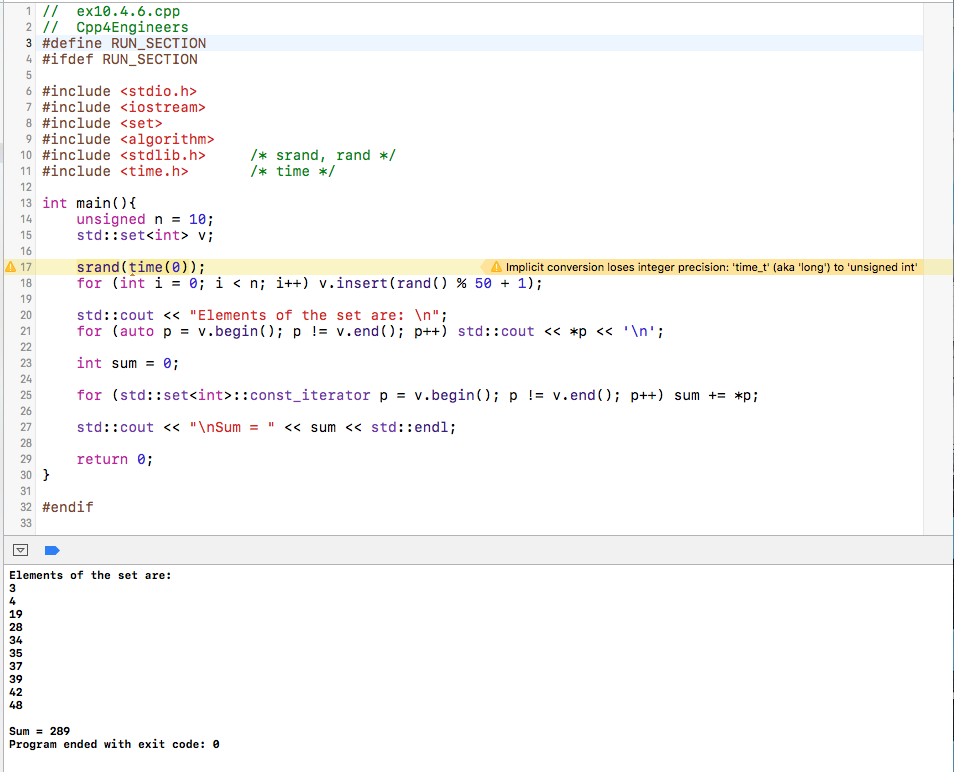
**Question 10.4.1**

The implementation and driver program for this question is in the file named “**ex10.4.1.cpp**”. Below is the output screen and the driver program.



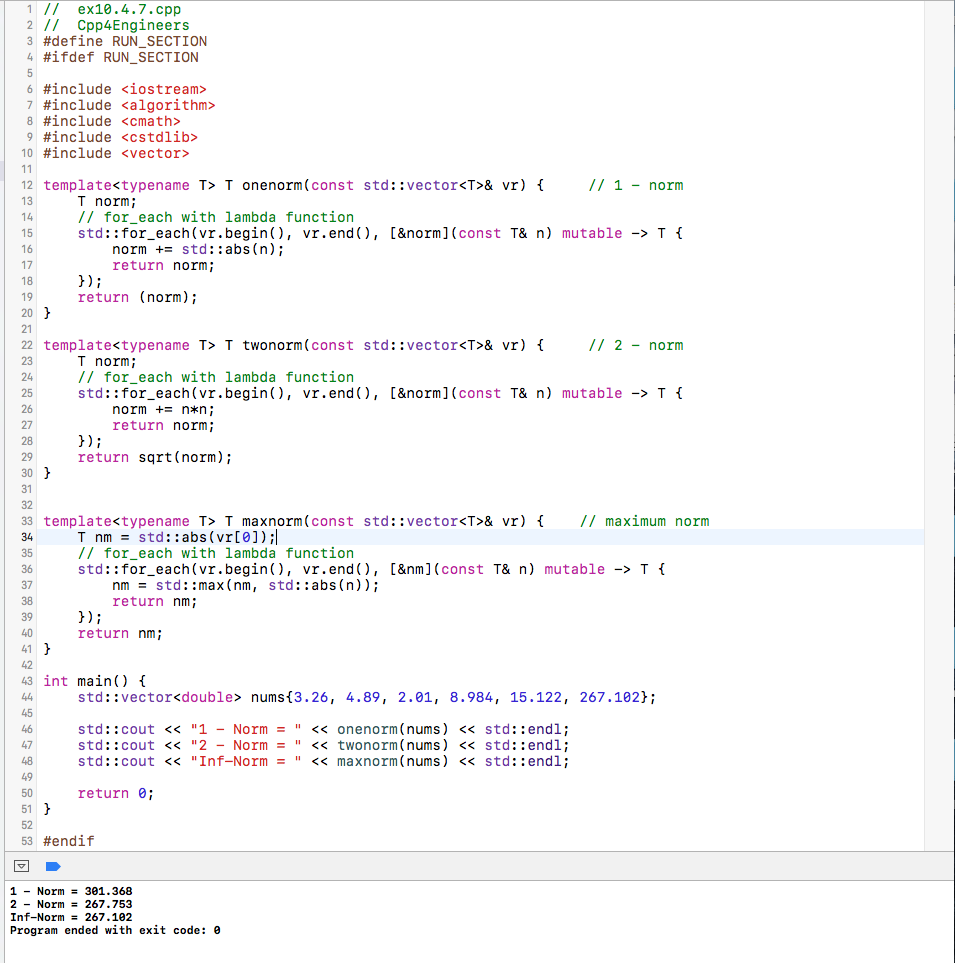
**Question 10.4.6**

The implementation and driver program for this question is in the file named “**ex10.4.6.cpp**”. Below is the driver program as well as the output screen.



**Question 10.4.7**

The implementation and driver program for this question is in the file named “**ex10.4.7.cpp**”. Below is the driver program as well as the output screen.



**References**

Yang, D. (2012). C and object-oriented numeric computing for scientists and engineers. New York: Springer.