MCC - CIS162AB - C++ Level I
P07 Using the Debugger - 10 points

The purpose of this assignment is to use the Visual C++ .Net debugger tool to figure out why the program below displays some very strange amounts regardless of what is entered by the user. The debugger tool can help programmers find such logic errors. If you find the bug before completing the assignment, don't correct it. Wait until after you have completed all the steps in this assignment.

This assignment should also help you better understand call-by-value, call-by-reference, and the scope of local variables.

The debugger allows the programmer to view the intermediate results stored in variables as the program is executed one step at a time or several lines of code at a time. When executing several lines at a time, the programmer must tell the debugger when to stop by setting a **breakpoint** on a specific line of code.

This handout walks you through the process of using the debugger tool using a program similar to P05-ex and P06-ex.

1. Startup Visual C++ and maximize the window size.

2. Create a new **Win32 Console Application projected named P07.**
   Add a new **Text file names output.txt**
   Add a new **C++ Source file named p07**.

3. **Copy and paste** the following source code into the source code file p07.cpp.

```
//P07 Using the Debugger -  Your Name
//
//   This program is used by customers to determine what the cost
//   of their order would be based on the price and quantity ordered.

#include <iostream>
using namespace std;

//Function Prototypes
void getPrice(double& price);             //call-by-reference
void getQuantity(int quantity);           //call-by-value

double calcCost(double cost, int quantity);
double calcTax(double subtotal, double taxRate);

//Declare the global constants
    const double TAX_RATE = 0.05;

void main()
{
//Declare the local constants
    const double SHIPPING = 10.00;

//Declare local variables
    int  quantity;
    double price, subtotal, salesTax, total;

//Set the decimal point to 2 positions
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);

    cout << "P07 - Your Name \n\n";

//Get and validate values
    getPrice(price);
    getQuantity(quantity);

//Calculate amounts
    subtotal = calcCost(price, quantity);
    salesTax = calcTax(subtotal, TAX_RATE);

    total = subtotal + salesTax + SHIPPING;

//Display the results
    cout << endl
         << "Price:    \t" << price     << endl
         << "Quantity: \t" << quantity  << endl
         << "Subtotal: \t" << subtotal  << endl
         << "Sales Tax:\t" << salesTax << " at " << TAX_RATE << endl
         << "Shipping: \t" << SHIPPING  << endl
         << "Total Due:\t" << total     << endl;

    cout << "\nThank you!\n\n";

    return;
} // end of main


//Function Definitions
void getPrice(double& price)
{
    do
    {
        cout << "Enter a value between $5 and $15.00 for the price: ";
        cin >> price;

    } while (price < 5 || price > 15);

    cout << endl;
    return;
}


void getQuantity(int quantity)
{
    do
    {
        cout << "Enter a value between 1 and 50 for the quantity: ";
        cin >> quantity;

    }while (quantity < 1 || quantity > 50);
    return;
```

```
}


double calcCost(double price, int quantity)
{
    double subtotal;
    subtotal = price * quantity;
    return (subtotal);
}


double calcTax(double subtotal, double taxRate)
{
    double amount;
    amount = subtotal * taxRate;
    return (amount);
}
//end of program
```

4. **Select Build P07** to compile and link the program.
   This should verify that everything was copied correctly.
   There should be zero errors and 1 warning. Ignore the warning for now.

5. **Execute** the program so that you can see the result of the bug.
   Click on **Debug** on the menu, and scroll down and select **Start Without Debugging.**

   - Enter 10 for the price.
   - A Run-time Debug Error will be displayed regarding the variable quantity.
   - Click on Ignore to continue.
     This message may be displayed up to 3 times during the execution.
     Click on Ignore each time.
   - Enter 10 for the quantity.
   - Eventually some very strange results should be displayed for quantity and other variables that store values calculated using quantity.
   - **Capture this output and save in output.txt**

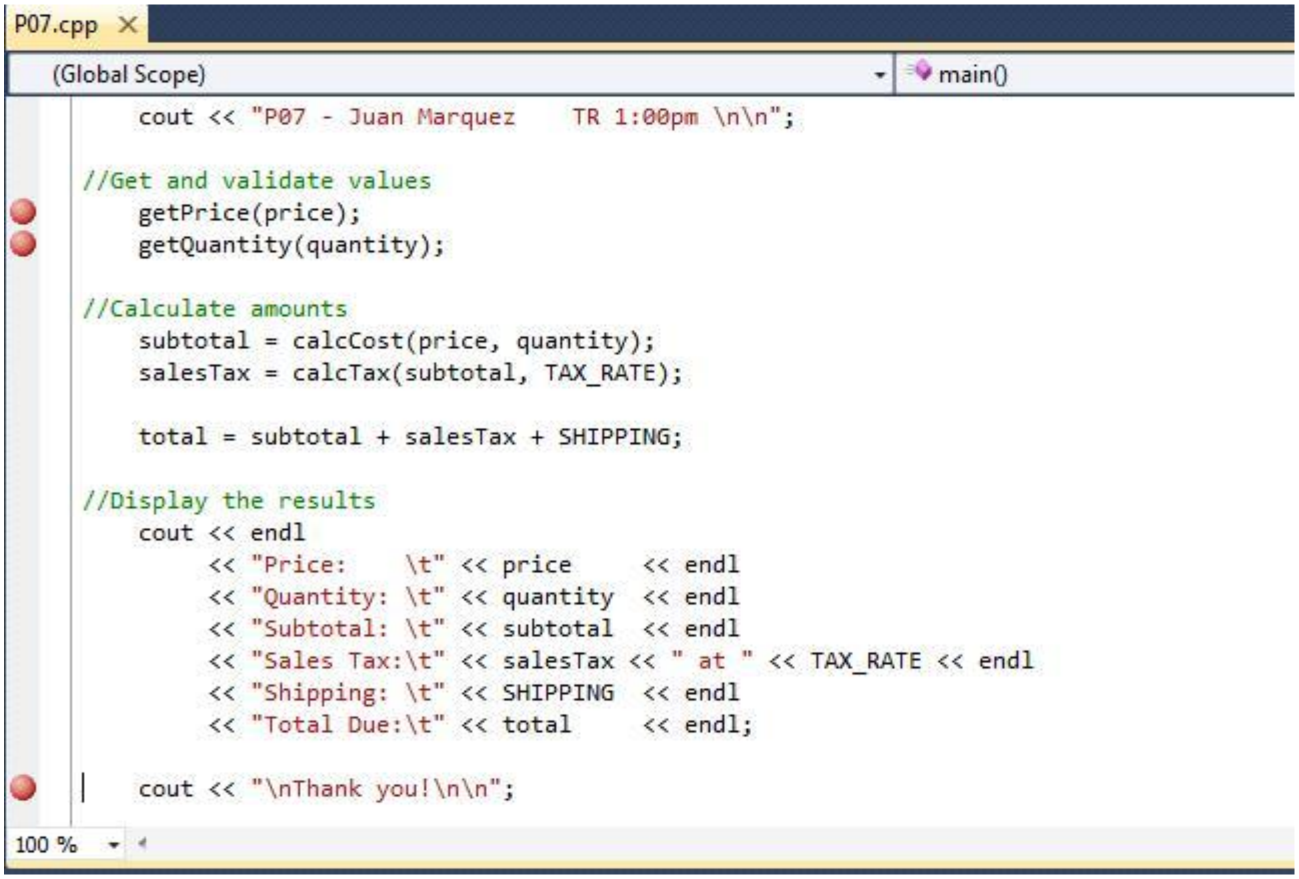
```
P07 - Juan Marquez    TR 1:00pm

Enter a value between $5 and $15.00 for the price: 10

Enter a value between 1 and 50 for the quantity: 10

Price:          10.00
Quantity:       -858993460
Subtotal:       -8589934600.00
Sales Tax:      -429496730.00 at 0.05
Shipping:       10.00
Total Due:      -9019431320.00

Thank you!

Press any key to continue
```

- Reminder: Capture this output and save in output.txt

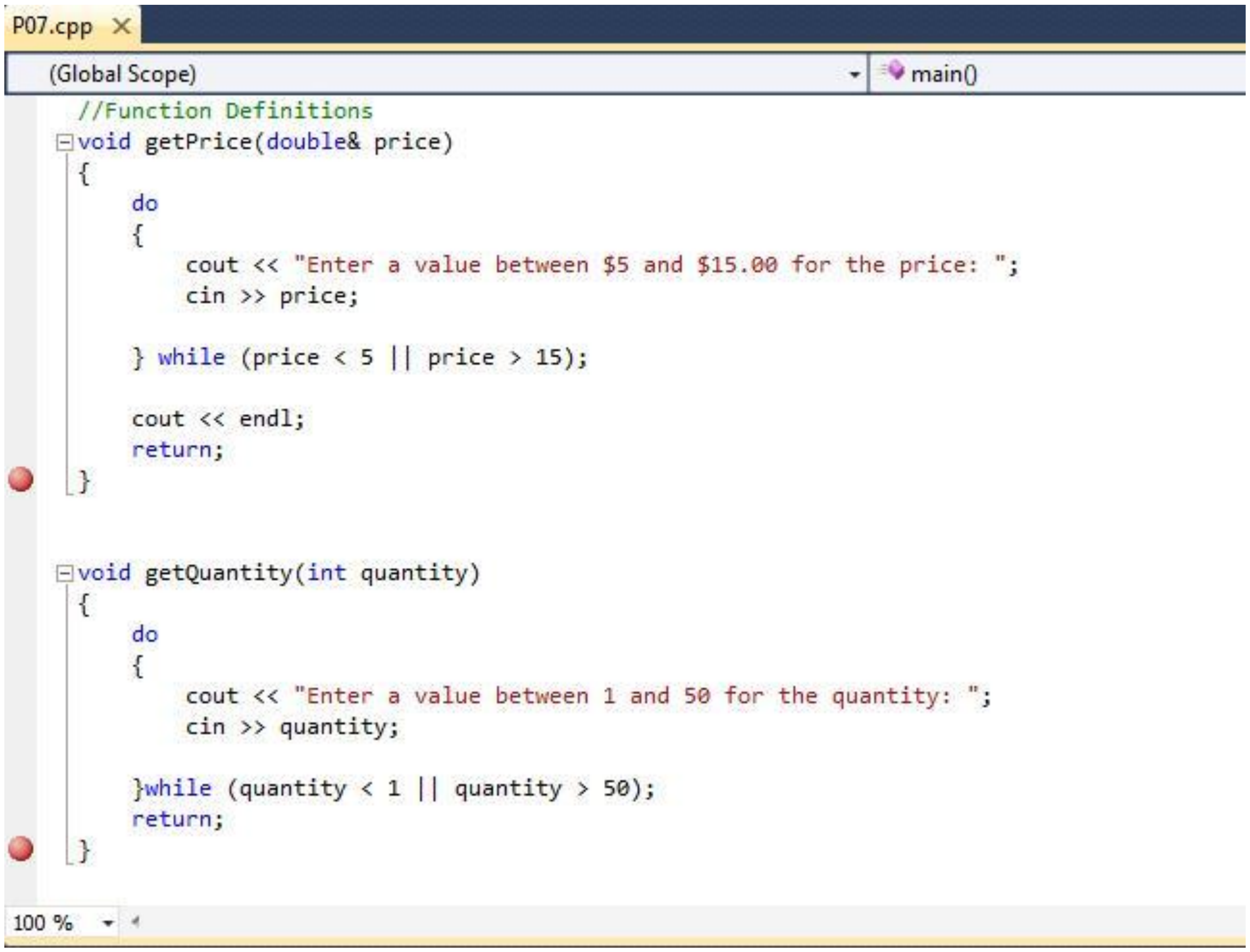
6. During the debugging process you may want to pause the execution of the program at a particular line code, so that you can observe the values in variables or to watch what happens as each line of code is executed. To cause the program to pause a **breakpoint** must be inserted.

- To set a breakpoint, place the mouse pointer in the margin area that appears as a border at the left edge of the Editor window and click by the line of code you desire the break. A large red dot will display in the margin.
- To remove a breakpoint go to the statement that has a breakpoint set, and click on the red dot located in front of the line code.
- **Set a breakpoint** at each of the following statements:

  - getPrice(price);
  - getQuantity(quantity);
  - cout << "Thank you";
  - on the closing brace } after the return statement in getPrice.
  - on the closing brace } after the return statement in getQuantity.



```
    cout << "P07 - Juan Marquez    TR 1:00pm \n\n";

//Get and validate values
    getPrice(price);
    getQuantity(quantity);

//Calculate amounts
    subtotal = calcCost(price, quantity);
    salesTax = calcTax(subtotal, TAX_RATE);

    total = subtotal + salesTax + SHIPPING;

//Display the results
    cout << endl
         << "Price:    \t" << price     << endl
         << "Quantity: \t" << quantity  << endl
         << "Subtotal: \t" << subtotal  << endl
         << "Sales Tax:\t" << salesTax << " at " << TAX_RATE << endl
         << "Shipping: \t" << SHIPPING  << endl
         << "Total Due:\t" << total     << endl;

    cout << "\nThank you!\n\n";
```



```
//Function Definitions
void getPrice(double& price)
{
    do
    {
        cout << "Enter a value between $5 and $15.00 for the price: ";
        cin >> price;

    } while (price < 5 || price > 15);

    cout << endl;
    return;
}


void getQuantity(int quantity)
{
    do
    {
        cout << "Enter a value between 1 and 50 for the quantity: ";
        cin >> quantity;

    }while (quantity < 1 || quantity > 50);
    return;
}
```

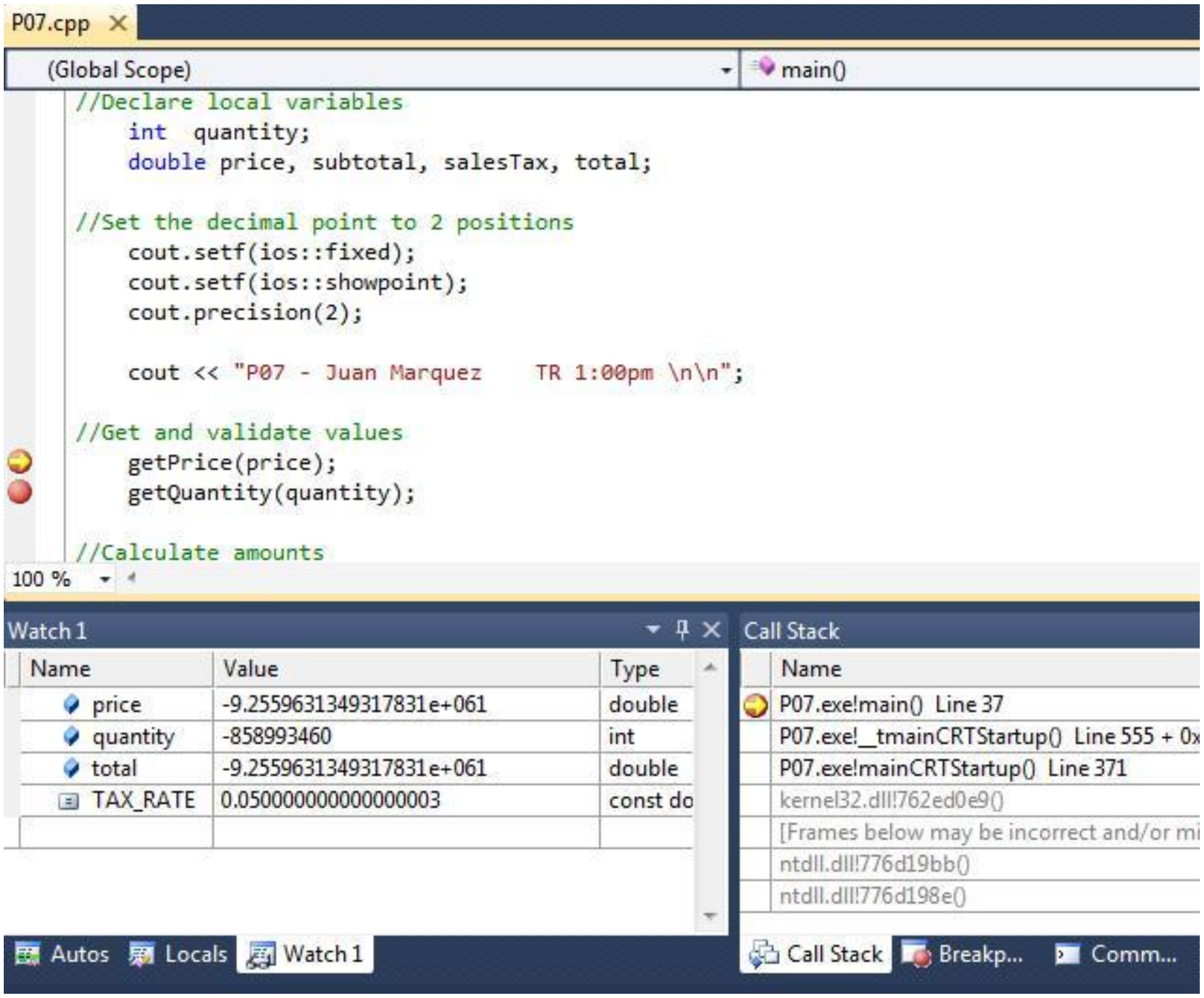
7. Run the program in debug mode.

- Click on **Debug** on the menu, and scroll down and select **Start Debugging.** The F5 function key can also be used.
- The **execution will stop** at the first breakpoint, which should be the function call getPrice(price).
- The debugger should open **three windows**.

  - Top left window is the source code.
  - The bottom right panel can display the Command Window, Call Stack, Breakpoints, etc.
  - The bottom left panel can display Auto, Local, and Watch variables (tabs listed at bottom).
  - Click on the **Watch 1** tab to bring it forward.
  - The Watch 1 panel is where a programmers enter variable names that they want to watch as the variables go in and out of scope and as their values change.
  - If these windows aren't opened, open the missing window by selecting the menu options:
    Debug -> Windows -> Locals
    Debug -> Windows -> Watch 1

8. **Set a watch** on the variables price, quantity, total, and TAX_RATE as follows:

- On the bottom left Watch 1 panel,
  click in the first empty cell under the column heading **Name**.
  Type in **price**.
- Click on the next empty cell. Type in **quantity**.
- Click on the next empty cell. Type in **total**.
- Click on the next empty cell. Type in **TAX_RATE**, which is a global variable.



```
//Declare local variables
    int  quantity;
    double price, subtotal, salesTax, total;

//Set the decimal point to 2 positions
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);

    cout << "P07 - Juan Marquez    TR 1:00pm \n\n";

//Get and validate values
    getPrice(price);
    getQuantity(quantity);

//Calculate amounts
```

Watch 1

Name	Value	Type
price	-9.2559631349317831e+061	double
quantity	-858993460	int
total	-9.2559631349317831e+061	double
TAX_RATE	0.050000000000000003	const do

Call Stack

Name
P07.exe!main() Line 37
P07.exe!_tmainCRTStartup() Line 555 + 0x
P07.exe!mainCRTStartup() Line 371
kernel32.dll!762ed0e9()
[Frames below may be incorrect and/or mi
ntdll.dll!776d19bb()
ntdll.dll!776d198e()

9. **Control the execution** of the program in debug mode using the function keys F5 and F11.
   F5 executes until the next breakpoint or end of program.
   F11 steps through the program one statement at time.

10. **Step through getPrice()**
    - Step into the getPrice function using **F11**.
    - Since price is the only local variable, the **list of local variables** should be reduced to price. On the **Watch list**, only price should have a value, and the others will have a message stating that they were not found (CXX0017: Error: symbol "quantity" not found). However, since TAX_RATE is global variable, its value is still displayed.
    - Skip to the next breakpoint by pressing **F5**. Stepping through the cout and cin procedures is very tedious.
    - Before reaching the next breakpoint, a value for the cin statement in getPrice must be entered. **Enter 10 for the price** and **press return.**
    - Click on the source code window to bring it to the foreground if it is not visible.
    - The **next breakpoint** is at end of the getPrice function.
    - Notice in the Watch window that price was actually assigned the value of 10.
    - Press **F11** to step back into main.
    - Notice that the value of 10 was returned to the local variable price defined in main().
    - Press **F11** to get ready to step into getQuantity.

11. **Step through getQuantity()**
    - Press **F11** to step into getQuantity.
    - Click on **Continue** for the Run-Time Error message.
    - Before reaching the breakpoint at end of the function, a value for quantity must be entered. **Enter 10 for the quantity** and **press return**.
    - Click on the source code window to bring it to the foreground if it is not visible.
    - Notice in the Watch window that the local variable quantity was assigned a value of 10, and that price is now out of scope.
    - Press **F11** to step back into main.
    - In the variable Watch window we should see that the local variable quantity in main does not have the value of 10 that was entered and visible in getQuantity(). In this case the value entered in the getQuantity function was not returned to main. So **the bug is returning the value from the function getQuantity**.

12. What is the error?
    How is the value in getQuantity() supposed to get back to main()?
    Include your explanation at the bottom of output.txt

13. **Complete debug session**
    - Press **F5** to reach the last statement in the program.
    - Click on **Continue** for the two additional Run-Time Error messages.
    - Press **F5** to end the debug session.
    - Any libaries (cout, etc) that you stepped into while in debug will need to have their source code files closed.

14. The values in variables can be changed during execution while in debug mode.
    - Press **F5** to get the debugger to Go again.
    - Press **F5** to execute the getPrice function.
    - Click on the console output window to bring it to the foreground if it is not visible.
    - **Enter 10 for the price.**

- Click on the source code window to bring it to the foreground if it is not visible.
- Press **F5** to return to main.
- Press **F5** to execute the getQuantity function.
- Click on **Continue** for the Run-Time Error message.
- Click on the console output window to bring it to the foreground if it is not visible.
- **Enter 10 for the quantity.**
- Click on the source code window to bring it to the foreground if it is not visible.
- At the end of getQuantity, press **F11** to return to main.

- On the bottom left window (variable window), **double click on the value of quantity.**
- **Replace the current value with a 10** and **press enter.**
- Press **F5** to complete the calculations.
- Even though a value has been entered for quantity, we must still
  click on **Continue** for the two additional Run-Time Error messages.
- Notice that the variable now has a valid value.
  So, the problem is that quantity is not getting back to main().
  The rest of the logic seems to work when quantity has a valid value.
- The statement pointer should be at the cout statement that displays thank you.
- **Click on the console output window** on the task bar to see the current output.
- Since we manually corrected the quantity, some valid output is now displayed on the output.
- Click on the source code window to bring it to the foreground.
- Press **F5** to end the debugging session.

15. Complete the assignment.
- Correct the bug in the program at this time.
- Execute the program again to test the correction.
- **Capture the correct output and save in output.txt**

```
P07 - Your Name

Enter a value between $5 and $15.00 for the price: 10

Enter a value between 1 and 50 for the quantity: 10

Price:          10.00
Quantity:       10
Subtotal:       100.00
Sales Tax:      5.00 at 0.05
Shipping:       10.00
Total Due:      115.00

Thank you!

Press any key to continue . . .
```

16. Submit the corrected P07.cpp and output.txt.