

CS-405 Secure Coding – Module 1 Assignment

Eric Slutz

Southern New Hampshire University

MODULE ONE ASSIGNMENT	3
OVERFLOW TESTS SCREENSHOT	3
UNDERFLOW TESTS SCREENSHOT	4
SUMMARY OF OVERFLOW DETECTION PROCESS.....	4

Module One Assignment

Overflow tests screenshot

```
Microsoft Visual Studio Debug Console
+ -

*****
*** Running Overflow Tests ***
*****
Overflow Test of Type = char
    Adding Numbers Without Overflow (0, 25, 5) = 125
    Adding Numbers With Overflow (0, 25, 6) = Numeric overflow has occurred!

Overflow Test of Type = wchar_t
    Adding Numbers Without Overflow (0, 13107, 5) = 65535
    Adding Numbers With Overflow (0, 13107, 6) = Numeric overflow has occurred!

Overflow Test of Type = short
    Adding Numbers Without Overflow (0, 6553, 5) = 32765
    Adding Numbers With Overflow (0, 6553, 6) = Numeric overflow has occurred!

Overflow Test of Type = int
    Adding Numbers Without Overflow (0, 429496729, 5) = 2147483645
    Adding Numbers With Overflow (0, 429496729, 6) = Numeric overflow has occurred!

Overflow Test of Type = long
    Adding Numbers Without Overflow (0, 429496729, 5) = 2147483645
    Adding Numbers With Overflow (0, 429496729, 6) = Numeric overflow has occurred!

Overflow Test of Type = __int64
    Adding Numbers Without Overflow (0, 1844674407370955161, 5) = 9223372036854775805
    Adding Numbers With Overflow (0, 1844674407370955161, 6) = Numeric overflow has occurred!

Overflow Test of Type = unsigned char
    Adding Numbers Without Overflow (0, 51, 5) = 255
    Adding Numbers With Overflow (0, 51, 6) = Numeric overflow has occurred!

Overflow Test of Type = unsigned short
    Adding Numbers Without Overflow (0, 13107, 5) = 65535
    Adding Numbers With Overflow (0, 13107, 6) = Numeric overflow has occurred!

Overflow Test of Type = unsigned int
    Adding Numbers Without Overflow (0, 858993459, 5) = 4294967295
    Adding Numbers With Overflow (0, 858993459, 6) = Numeric overflow has occurred!

Overflow Test of Type = unsigned long
    Adding Numbers Without Overflow (0, 858993459, 5) = 4294967295
    Adding Numbers With Overflow (0, 858993459, 6) = Numeric overflow has occurred!

Overflow Test of Type = unsigned __int64
    Adding Numbers Without Overflow (0, 3689348814741910323, 5) = 18446744073709551615
    Adding Numbers With Overflow (0, 3689348814741910323, 6) = Numeric overflow has occurred!

Overflow Test of Type = float
    Adding Numbers Without Overflow (0, 6.80565e+37, 5) = 3.40282e+38
    Adding Numbers With Overflow (0, 6.80565e+37, 6) = Numeric overflow has occurred!

Overflow Test of Type = double
    Adding Numbers Without Overflow (0, 3.59539e+307, 5) = 1.79769e+308
    Adding Numbers With Overflow (0, 3.59539e+307, 6) = Numeric overflow has occurred!

Overflow Test of Type = long double
    Adding Numbers Without Overflow (0, 3.59539e+307, 5) = 1.79769e+308
    Adding Numbers With Overflow (0, 3.59539e+307, 6) = Numeric overflow has occurred!
```

Underflow tests screenshot

```
Microsoft Visual Studio Debu...
*****
*** Running Underflow Tests ***
*****
Underflow Test of Type = char
    Subtracting Numbers Without Underflow (127, 25, 5) = 2
    Subtracting Numbers With Underflow (127, 25, 6) = Numeric underflow has occurred!

Underflow Test of Type = wchar_t
    Subtracting Numbers Without Underflow (65535, 13107, 5) = 0
    Subtracting Numbers With Underflow (65535, 13107, 6) = Numeric underflow has occurred!

Underflow Test of Type = short
    Subtracting Numbers Without Underflow (32767, 6553, 5) = 2
    Subtracting Numbers With Underflow (32767, 6553, 6) = Numeric underflow has occurred!

Underflow Test of Type = int
    Subtracting Numbers Without Underflow (2147483647, 429496729, 5) = 2
    Subtracting Numbers With Underflow (2147483647, 429496729, 6) = Numeric underflow has occurred!

Underflow Test of Type = long
    Subtracting Numbers Without Underflow (2147483647, 429496729, 5) = 2
    Subtracting Numbers With Underflow (2147483647, 429496729, 6) = Numeric underflow has occurred!

Underflow Test of Type = __int64
    Subtracting Numbers Without Underflow (9223372036854775807, 1844674407370955161, 5) = 2
    Subtracting Numbers With Underflow (9223372036854775807, 1844674407370955161, 6) = Numeric underflow has occurred!

Underflow Test of Type = unsigned char
    Subtracting Numbers Without Underflow (255, 51, 5) = 0
    Subtracting Numbers With Underflow (255, 51, 6) = Numeric underflow has occurred!

Underflow Test of Type = unsigned short
    Subtracting Numbers Without Underflow (65535, 13107, 5) = 0
    Subtracting Numbers With Underflow (65535, 13107, 6) = Numeric underflow has occurred!

Underflow Test of Type = unsigned int
    Subtracting Numbers Without Underflow (4294967295, 858993459, 5) = 0
    Subtracting Numbers With Underflow (4294967295, 858993459, 6) = Numeric underflow has occurred!

Underflow Test of Type = unsigned long
    Subtracting Numbers Without Underflow (4294967295, 858993459, 5) = 0
    Subtracting Numbers With Underflow (4294967295, 858993459, 6) = Numeric underflow has occurred!

Underflow Test of Type = unsigned __int64
    Subtracting Numbers Without Underflow (18446744073709551615, 3689348814741910323, 5) = 0
    Subtracting Numbers With Underflow (18446744073709551615, 3689348814741910323, 6) = Numeric underflow has occurred!

Underflow Test of Type = float
    Subtracting Numbers Without Underflow (3.40282e+38, 6.80565e+37, 5) = 0
    Subtracting Numbers With Underflow (3.40282e+38, 6.80565e+37, 6) = Numeric underflow has occurred!

Underflow Test of Type = double
    Subtracting Numbers Without Underflow (1.79769e+308, 3.59539e+307, 5) = 9.9792e+291
    Subtracting Numbers With Underflow (1.79769e+308, 3.59539e+307, 6) = Numeric underflow has occurred!

Underflow Test of Type = long double
    Subtracting Numbers Without Underflow (1.79769e+308, 3.59539e+307, 5) = 9.9792e+291
    Subtracting Numbers With Underflow (1.79769e+308, 3.59539e+307, 6) = Numeric underflow has occurred!

All Numeric Overflow / Underflow Tests Complete!
```

Summary of overflow detection process

The addition numeric overflow process was simple. I took the max possible value for that data type. I then subtracted the increment amount from that total. That way, when looking through and incrementing the number, the next step is checked to see if the addition will result in an overflow. The subtraction numeric overflow process took more work and research. My original solution was based on the solution from addition, but I was not able to get all site update to work

that way. After studying the code some more, I was able to find a solution by using the max for that particular data type, the decrement amount, and the number of steps. With this I was able to determine how many steps it would take before the value was overthrown and compare that to the number of steps passed in.