**LAB:  Advanced MSVS Debugger example using a real-life network failure**

Some of the text and code used in this exercise was **borrowed  *Deep C Programming: Deep C Secrets*, by Peter Van Der Linden (SunSoft Press, Prentice Hall, 1984), pp. 38-39.** The program that will be used in this exercise is called "**att\_bug.cpp**", which is an included document for the experiments of this lab.

The code is based on a program that caused a major disruption of AT&T phone service throughout the U.S. This failure made national news at the time and is one of the best-known software failures.

AT&T's nationwide network was mostly unusable for about nine hours starting on the afternoon of January 15, 1990. Telephone exchanges, also known as "switching systems", are computer systems that allow multiple phone calls to be transferred concurrently over one wire. This type of software was running on the hardware controlling the AT&T network. The failure of the network was due to the bug that will be illustrated in this example; The bug led to the first major network problem in AT&T's 114-year history. The bug caused a problem that led to a chain-reaction that spread across the network, bringing down AT&T's entire long distance network. More information on the bug can be found in the January 22, 1990 issue of *Telephony* magazine.

You should be able to understand the logic of the code (**att\_bug.cpp**). Certain **values are used to call various functions which initialize** certain **variables** **to make the system execute correctly**. Different initialization functions are called depending on the values of the initialized variables. A **switch** statement, **if**statements, and **break**statements are used to implement the correct initialization logic. In this simplified example, all **you need to know is that both global integer variables,** **val1** and **val2**, **in the program must be set to non-negative values before** **the** **function route\_that\_call()** is called.

**Testing Section:**

**To use the code in this exercise, simply compile and execute it,  and enter 3 integer values.** **The program will print a message stating whether "the phone system" is OK or crashes for the 3 integer values you entered. The program will repeatedly ask for 3 more values until it gets to the end of file or invalid inputs.**

**Step 1: Compile and execute the program, att-bug.cpp, with various inputs and see what happens.**

**Step 2: Study the logic of the code.  Answer the following questions:**

***Question 1:***For what values does the function **init\_values\_x1()** get invoked?

***Question 2:***For what values does **init\_values\_part1A()**get called?

***Question 3:***For what values does **init\_values\_part1B()** get called?

***Question 4:***For what values does **init\_values\_part2()** get called?

***Question 5:***For what values does **init\_default()**get called?

**Step 3: Find values for** **i**, **x** **and** **y** **that make the phone network crash? Hint: Try each set of data values you identified in Step 2 to make the program perform each of the possible initializations.  Together these sets of values are good test data, since each one causes the program to behavior differently.  They cover all the behavior caused by the different initialization values.**

***Question 6:***  What set of 3 values (**i,x,y**) caused the phone network to crash?

**Debugger Section:**

Now that you know what data is causing the crash, you may still not know why those data values cause the problem. In the steps that follow you will use the **MSVS 2010 debugger**and answer the following questions.  **Build** the program with the **Build Solution option** and then carry out the following steps.

**Step 4: Put a breakpoint at the start of the function, phone\_network(). Execute the program, by choosing "Debug" and then "Start Debugging".**  
***Question 7:***After entering the 3 values, what is displayed when the execution of the program stops at the breakpoint you set at the line that contain the name of the function?

**Step 5:  Type "Ctrl-F10" to resume execution.**

***Question 8:***  Does the program stop at the breakpoint again?

***Question 9:***  How do you remove the breakpoint you set?

**Step 6: If you removed the breakpoint, put it back. Execute the program again. When you reach the breakpoint, choose "F11".  This executes the program one line at a time. Execute "F11" a few times to see how "F11" works. Type "Ctrl-F10" to continue on until the breakpoint is reached again.**

**Now that you are back at the initial breakpoint again, do the same thing but this time choose "F10" instead of "F11" to go to the next line. Do this a few times to observe what happens.**

***Question 10****:*What differences in the execution of the **MSVS** commands, **"F11"** and **"F10",** did you observe?

**Step 7:  Using the same breakpoint from Step 6, run the "F11" command to step "into" a function.  Now enter "Shift-F11".  The "Shift-F11" command terminates the execution of the function and pauses.**  
  
***Question 11:*** Where does execution stop when you enter "**Shift-F11**"?

**Step 8:  Using the same breakpoint, stop the execution process by choosing "Stop Debugging" and then restart the execution process by choosing "Start Debugging"**.

**Now, in the** "**Watch**" **window underneath the code window, enter the variables**"**val1**" **and** "**val2**"**per each row under Name.** **Once you've** **done this, run the** "**F10**"**command a few times by typing** "**F10**"**repeatedly. Then type** "**Ctrl-F10**" **to run the**"**continue**"**command.**

***Question 12:***  What does **MSVS** do with "**Watch**" variables?

**Step 9:  OK, about that bug.... Set a breakpoint at the line that contains  phone\_network() or at the switch statement. Enter the 3 data values for i, x, and ythat caused the phone network to crash. From the breakpoint, use the "F10" command to step through the program.**

***Question 13:***  Did you observe anything that surprises you about the behavior of the program? Was the execution of any lines skipped that you thought should have been executed?

***Question 14:***  Can you explain the behavior you observed?