University of Washington
Department of Electrical Engineering

Name_____

1. Write a simple C program to compute the price of a trip to Europe this summer for you and 8 of your best friends. The cost per person is $977.50 each with a 5% discount and a 9.5% sales tax. Be certain to fully annotate your design.

2. The following C program is to be designed to ask the user how many friends will be traveling with them then will tell them how much the total bill will be.
   a. Complete following program to compute and display the total bill based upon the number of travelers, the cost, and your code from problem 1.
   b. Fully annotate the code.
   c. Correct any poor coding or style practices.

```
#include<stdio.h>

main()
{
    int travelers;
    float totalCost = 0.0f;

    YOUR CODE HERE

    printf("how many people will be traveling: ");
    scanf("%d", &travelers);

    YOUR CODE HERE TOO

    return;
}
```

3. a. Use your debugger to identify the problems with the following C program. See the short tutorial at the end of this assignment.
   b. Correct the problems you have identified in part a.
   c. Set the specified breakpoints, rerun the program, using your debugger, stop at each breakpoint, inspect, and record the values of each of the variables indicated.

```
/*
* Working with Variables
*/
#include <stdio.h>

// a global variable
int anInt0 = 9;

int main(void)
{
	// some local variables
	int anInt1 = 8;
	int anInt2, anInt3 anInt4;

	int anAnswer = 0;

	//  break point 0  inspect anAnswer,anInt0,anInt1
	anAnswer = anInt0 + anInt1;
	//  break point 1  inspect anAnswer,anInt0,anInt1

	//  print the answer followed by a new line \n
	printf("the answer is: %d\n", anAnswer);

	anAnswer = anInt0 * anInt1

	//  print the answer followed by a new line \n
	printf("the answer is: %d\n", anAnswer);

	//  break point 2  inspect anAnswer,anInt2
	anAnswer = anAnswer + anInt2;
	//  break point 3  inspect anAnswer,anInt2

	//  print the answer followed by a new line \n
	printf("the answer is: %d\n", anAnswer);

	anAnswer = anInt1 + anInt3;

	//  print the answer followed by a new line \n
	printf("the answer is: %d\n", anAnswer);

	//  break point 4 anAnswer, inspect anInt0, anInt1, anInt4
	anAnswer == anInt0 + anInt1 + anInt4;
	//  break point 5 inspect anAnswer, anInt0, anInt1, anInt4


	//  print the answer followed by a new line \n
	printf("the answer is: %d\n", anAnswer);

	//  print variables followed by a new line \n
	printf("the variables are: %d %d %d\n", anInt2, anInt3, anInt4);

	return 0;
}
```

Debugging on the PC

Learning to use a debugger is an invaluable skill to develop if one is planning to do any serious software development work. This is independent of whether one develops embedded or desktop applications.

From Wikipedia…

> *The terms "bug" and "debugging" are both popularly attributed to Admiral Grace Hopper in the 1940s. While she was working on a Mark II Computer at Harvard University, her associates discovered a moth stuck in a relay and thereby impeding operation, whereupon she remarked that they were "debugging" the system. However the term "bug" in the meaning of technical error dates back at least to 1878 and Thomas Edison. Further "debugging" seems to have been used as a term in aeronautics before entering the world of computers.*

As our first step into learning such tools, we will use the tools on the desktop. The *Visual Studio* development environment provides some rather effective tools that help you to debug your code running on the PC platform. Some useful features/abilities are:

- breakpoints
- single-stepping through or over code
- inspecting the contents of local/global variables
- inspecting and change the values of variables during runtime

For each of the above items, there are many ways to perform the desired action. A brief summary on how to do each one is as follows:

Breakpoints – To set a breakpoint, move the cursor to the main window left hand border next to the desired line of code and left-click your mouse button. Alternately, you can place your cursor on the desired line of code, select right mouse and make a selection from the menu that pops up. Personally, I prefer to 'run to cursor'.

Bear in mind that a breakpoint is *not* the place in your debugging process where you throw up your hands and go for a beer.

Single-stepping – To single-step ***over*** each instruction in the code, press the F10 key. To step ***into*** a function (rather that over), press the F11 key. Note that the code must be stopped (at a breakpoint, or before *main( )* has executed) to single-step through the code.

Like breakpoints, single-stepping is not the process of slowly sneaking out of the lab when you've reached a ~~breakpint~~ breakpoint.

Local/Global Variables – To inspect local/global variables, you can either look at the display in the bottom pane of the IDE which will display all of the variables (and their values) in the current context.

Alternately, select the variable that you wish to look at. Then, using the right mouse button, select either ***Add Watch*** or ***Quick Watch*** from the menu choices that appear. Explore a bit to see what the differences are.