Department of Electrical and Computer Engineering
University of Massachusetts Dartmouth

ECE160: Foundations of Computer Engineering I (Spring 2023)
Instructor: Dr. Liudong Xing

LAB # 7

(Relevant Lecture: #14 & #15)

Monday, March 13 (L1) and Wednesday, March 15 (L2)

OBJECTIVES

- To learn how to write user-defined functions (L#14, L#15)
- To practice debugging and how to use the debugger in Microsoft Visual Studio

SUBMISSION REQUIREMENT

1. Please follow "Submission Guidelines" in the lab section of the course website to submit your solution (program files) to the class M: drive by **5pm, Wednesday, March 15**.
2. Suggested format for naming your solution files: lab#-your last name-p#.cpp
   For example: lab7-xing-p1.cpp for problem 1; lab7-xing-p2.cpp for problem 2; …

EXERCISES

Part I: Functions

1. **There are compilation errors in the following program. Correct the errors and run the program.**

```
#include <stdio.h>
float average_2(int num1, int num2)
void main(void)
{
    int n1;
    int n2;
    float avg_num;
    printf("Enter  two integers:\n");
    scanf_s("%d%d", &n1, &n2);

    avg_num = average_2(n1, n2);
    printf("The average is %f\n", local_average);
}

float average_2(int num1, int num2);
{
    float local_average;
    local_average = (n1 + n2)/2.0;
    return local_average;
}
```

Example Runs to Test your Program:
Input 19 and 7 from the keyboard, "The average is 13.000000" is displayed

2. (**Modifying exercise for Lab #5 Problem #4**) The program below reads 2 integer numbers from the keyboard and adds the two numbers. If the sum of the two numbers is an even number, it outputs "The sum is an even number". If the sum of the two numbers is an odd number, it outputs "The sum is an odd number". Modify the program according to the following requirement: Instead of doing the addition and parity checking in the main() function (highlighted in yellow), do both in a function called ParityCheck you need to define.

```
#include <stdio.h>
void main(void)
{
    int a;
    int b;
    int sum;
    a = 0;
    b = 0;
    sum = 0;

    printf("Please input two integer numbers:\n");
    scanf_s("%d%d", &a, &b);

    sum = a+b;
    if ((sum % 2) == 0)
        printf("The sum is an even number");
    else
        printf("The sum is an odd number");
}
```

Example Runs to Test your Program:

- Input 13 and 9 from the keyboard, "The sum is an even number" is displayed
- Input 10 and 17 from the keyboard, "The sum is an odd number" is displayed

Part II: Debugging

Debugging: the process of looking for and correcting errors that cause your program to behave unexpectedly. In the case of errors:

A. Look for the common errors: think globally, look at the whole program, and ask yourself questions such as:
   a. Did I type anything wrong? For example, was printf typed print?
   b. Did I use and follow C punctuation properly? For example, void main(void) being typed void main(void);
   c. Are my parentheses and braces in pairs?
   d. Did I use nested comments? Did I miss the closing */ in the comments?
   e. Did I use ; to end the statements?

B. Do not rely on the C compiler to locate errors.
   For example: you may get 30 error messages by just missing the closing */ in one of the comments at the beginning of the program. In other words, you may be able to eliminate dozens of error messages by correcting a single minor error.

C. Be as independent as you can be, seek help only when you are completely stuck.

3. **After reading the above information, debug and run the following program (hint: there are 8 errors in the program, some lines may contain more than one error):**

```
/* A debugging example*/
 #<include stdio.h>;
void main(void);
(
    printf('DEBUGGING EXAMPLE:\n');
    printf("This is an example of debugging. By following the step-by-step procedure ")
    printf("that we describe in this application example, you will begin to develop the );
    printf{"skills you need for successful and efficient debugging."};
}
```

4. **Learn the debugging techniques in Microsoft Visual Studio by going through the following steps, and submit the program files in step 2) to your folder in M: drive.**

   1) Open/create the project as usual
   2) Enter the .cpp file

```
#include "stdio.h"
int main(void)
{
    float celsius=0;
    float fahrenheit=0;
    float temp=0;
    float constant=0;
    printf("This program converts Celsius  to Fahrenheit. \n");
    printf("Please enter a Celsius temperature. \n");
    scanf_s("%f", &celsius);
    constant = 9.0/5.0;
    temp = constant*celsius;
    fahrenheit = temp + 32;
    printf("The temperature in Fahrenheit is: %f\n", fahrenheit);
    return 0;
}
```

   3) Compile the file using *Build Solution*.
   4) To run the program using the debugging feature, please follow the following steps. Note that as you debug, a window appears at the bottom having the name and the values of all the variables. As you execute each line in the program you will see the value of the corresponding variable changed. Observing these values, you may find any kind of logical errors in the program.
      a) Go to Debug and use the option Step Into (or F11 key). This starts the debugging from the first line of main()
      b) Now to execute each and every step of the program, go to Debug and select Step Over option (or F10 key).
      c) Continue F10 key till the scanf_s() line is executed. Enter the Celsius value at the command prompt.
      d) Continue debugging using the F10 key till the program exits successfully.

   Note: to stop the debugging any time, press Shift and F11 keys.

5) To learn the following debugging functions in Visual Studio:

- Inserting breakpoints
- Running to a breakpoint
- Value examination
- Breakpoint removal
- Stop debugging

a) To set a breakpoint on any line of the code, use the following steps:
   - We have to specify the line at which we want to set the breakpoint. Let us set the breakpoint on the line which starts with the calculations part, i.e., `temp=constant*celsius;`
   - Highlight the line and right click the mouse and select *Breakpoint* option and then *Insert Breakpoint*
   - Now the breakpoint is set.

b) To execute the program, use *Debug* → *Start Debugging*
   - Now instead of debugging from the first line in main(), it starts the debugging from the specified line.
   - To continue the debugging, use the Step Over (F10 key) option.

c) To stop the debugging, use *Debug* → *Stop Debugging*

d) To remove the breakpoint, go to *Debug* and select the option *Delete All Breakpoints*

e) You can also disable all the breakpoints using the option *Debug* → *Disable All Breakpoints*. To enable them, use *Debug* → *Enable All Breakpoints*