Programming Fundamentals for Software and Computer
Lab 1 – Fall 2021

Department of Electrical & Computer Engineering
University of Calgary
M. Moussavi PhD, P.Eng

Acknowledgment: This lab contains material written for previous versions of this course by Dr. M. Moussavi, and Dr. S. Norman.

Instructors:

- Dr. Mahmood Moussavi (B01, B02)
- Dr. Maan Khedr (B03, B04)

In This Lab You Can Work with a Partner (Groups of three or more are NOT allowed):

Working with a partner usually should give you the opportunity to discuss some of details of the topics and learn from each other. Also, it will give you the opportunity to practice one of the popular methods of program development called, **pair-programming**. In this method, which is normally associated with "Agile Software Development" technique, two programmers work together normally on the same workstation (you may consider a zoom session for this purpose). While one partner, the driver, writes the code, the other partner, acts as observer, looks over his/her shoulder making sure the syntax and solution logic is correct. Partners should switch roles frequently in a way that both of them have equivalent opportunity to practice both roles.

If you decided to work with a partner:

- **choose a partner that is in you lecture section. For example, students in lecture section 1 (L01 – Dr. Moussavi) cannot have a partner from lecture section 2 (L02 – Dr. Khedr). In other words, here are the lab sections that can partners:**
    - **B01 and B02 can be become partners**
    - **B03 and B04 also can become partners.**
- **Please submit only one lab report with both names. Submitting two lab reports with the same content will be considered as copies and plagiarism.**

Objectives:

This lab consists of several exercises, mostly designed helping you to:

- Practice drawing AR diagrams
- Understand simple application of pointers in C

Please heed this advice:

- You are strongly advised to study lab documents before coming to your scheduled lab sessions and at least complete the first few exercises at home.

- **Some exercises in this lab and future labs will not be marked.** Please do not skip them, because these exercises are as important as the others in learning the course material.

- In courses like ENSF 337 some students skip directly to the exercises that involve writing code, postponing the diagram drawing until later. That's a bad idea for two reasons:

    - Drawing diagrams is an important part of learning how to visualize memory used in C and C++ programs. If you do diagram-drawing exercises at the last minute, you won't learn the material very well.

    - If you do the diagrams first, you may find it easier to understand the code-writing exercises, so you may be able to finish them more quickly.

Due Dates:

B01	**Thursday Sept 30, 2:00 PM**
B02	**Thursday Sept 30, 2:00 PM**
B03	**Thursday Sept 30, 2:00 PM**
B04	**Tuesday Sept 28, 11:00 AM**

Marking Scheme:

You shouldn't submit anything for the exercises that are not marked.

Exercise	Marks
A	8
B	8
C	8
D	5 marks for part 2
E	4
F	3

Total: 36 marks

Exercise A - Projectile Time and Distance Calculator

Read This First

In physics, assuming a flat Earth and no air resistance, a projectile launched with specific initial conditions will have a predictable range (maximum distance), and a predictable travel time.

The range or maximum horizontal distance traveled by the projectile can be approximately calculated by:

$$\boldsymbol{d} = \frac{\boldsymbol{v}^2}{\boldsymbol{g}} \ \mathbf{sin}\ (\mathbf{2\Theta})$$

Where:

*g is gravitation acceleration (*9.81 m/s^2)
θ: the angle at which the projectile is launched in degrees
v: the velocity at which the projectile is launched
d: the total horizontal distance travelled by the projectile

To calulate the projectile travel time (t), when it reaches the maximum horizontal distace the followig formaula can be used :

$$\boldsymbol{t} = \frac{\boldsymbol{2vsin\Theta}}{\boldsymbol{g}}$$

In this exercise you will complete a given C source file called `lab1exe_B.c` that prompts the user to enter a projectile's initial launch velocity (v) and displays the table of maximum horizontal distance and travel time for the trajectory angles of 0 to 90 degrees.

What to Do:

First, download file `lab2exe_A.c` from D2L. In this file the definition of function main and the function prototypes for four other functions are given. Your job is to complete the definition of the missing functions as follows:

Function create_table: which is called by the main function, receives the projectile initial velocity and displays a table of the projectile's maximum travel distance (d) and time (t), for trajectory angles of 0 to 90 (degrees), with increments of 5 degrees. Here is the sample of the required table:

```
 Angle          t            d
(deg)        (sec)         (m)
0.000000     0.000000     0.000000
5.000000     1.778689     177.192018
10.000000    3.543840     349.000146
```

You don't have to worry about the format or the number of digits after the decimal point. The default format is acceptable.

Function projectile_travel_time: receives two double arguments, the trajectory angle (θ), and the initial velocity (v) and returns the projectile travel time (t).

Function projectile_travel_distance: receives two double arguments, the trajectory angle (θ), and the initial velocity (v) and returns the projectile maximum horizontal distance (d).

Function degree_to_radian: receives an angle in degrees and converts to radian. This function is needed, because the C library function `sin` needs its argument value to be in radian.

Notes:

- To use the C library function `sin`, you need to include header file `math.h`.
- When compiling from command line, if you are using Cygwin, Geany, or XCode, use `–lm` option to link the math library:

  ```
  gcc -Wall –lm lab1exe_B.c
  ```

- For constant values of π, and gravitation acceleration, `g`, the following lines are already included in the given file:

  ```
  #define 3.141592654
  #define G 9.8
  ```

What to Submit:

Submit the copy of your source code (program) and its output as part of your lab report in PDF format.

Exercise B: Drawing AR Diagrams for a Simple C Program

Read This First:

This is a very simple exercise on activation records that we discussed during the lectures. To learn more about activation records, please read the set of slides posted on the D2L, called: `"03_ActivatonRecord.pdf"`.

What to Do

Download file `lab2exe_B.c` from D2L. Then, use pencil and paper to make memory diagrams for points one, two and three in the program.

What to Submit: Submit your AR diagrams, as part of your lab report.

Exercise C – Introduction to Pointers

Read This First

This is an important exercise in ENSF 337*.* If you don't become comfortable with pointers, you will not be able to work with C. Spend as much time on this exercise as is necessary to understand exactly what is happening at every step in the given program.

What to do

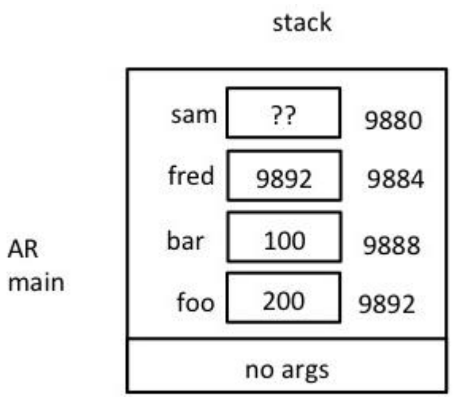
Download the file `lab2exe_C.c`. Trace through the execution of the program to determine what the output will be.

Now assume the following addresses for variables:

```
 sam   9880
 fred  9884
 bar   9888
 foo   9892
```

Draw a set of AR diagrams for points two through five and use the given address numbers as values of the pointers (don't use arrow notation in this exercise).

To understand how to draw these diagrams the solution for point one is given in the following figure:



After you completed the exercise, compile lab2exe_C.c and check your predicted output against the actual program output.

What to Submit:

Submit your diagrams as part of your lab report.

Exercise D: Pointers as function arguments

What to do – Part I

First copy the file `lab2exe_D1.c` from D2L. Carefully make diagrams for point one in this program using "**Arrow Notation**" as we discussed during lectures (you don't need to use made-up addresses). Then compare your solution with the posted solution on D2L.

There is nothing to submit for this part

What to do – Part II

Now download the file `lab2exe_D2.c` from D2L and draw AR diagram for point one in this file, for pointers using **arrow notation**.

What to Submit:

Submit the AR diagram for part II as part of your lab report.

Exercise E: Using pointers to get a function to change variables

What to do

Make a copy of the file `lab2exe_E.c` from D2L. If you try to compile and run this program it will give you some warning because the definition of function `time_convert` is missing.

Write the function definition for `time_convert` and add code to the main function to call `time_convert` before it prints answers.

What to Submit:

Submit the copy of your source code, `lab2exe_E.c` and your program's input and output as part of your lab report in PDF format.

About the function interface comment

The line minutes_ptr and seconds_ptr point to variables, appears in the REQUIRES section for the time_convert function. You might think that's useless information, but it's not. It is quite possible for a pointer argument to contain a garbage address, or a null pointer value, and in either of these cases the program would almost certainly crash. (You'll learn about null pointers later in the course.)

Exercise F: More on `scanf`

Read This First:

`scanf` returns an integer number which indicates the number of input items that `scanf` has successful read from the keyboard. For example in the following code segment `scanf` is supposed to read three integer numbers and put them into the variables a, b, and c.

```
int a, b, c, nscan;
printf("Please Enter three integer values: ");
nscan = scanf("%d%d%d", &a, &b, &c);

if (nscan != 3) {
   printf("Error: invalid input(s). I quit.\n\n");
   exit(1);
}
```

To detect input error after calling `scanf`, a program must check the *return value* from `scanf`. If user enters invalid input, the return value of the `scanf`, `nscanf`, will have a value of zero, one, or two depending on how many of the inputs were properly entered. In this example, if `nscanf` is not equal to 3 the program will terminate using the exit() library function.

In general, scanf returns the number of items that it reads successfully, or returns EOF. EOF is a negative constant defined in <`stdio.h`>.The value EOF means that `scanf` reached the end of an input file or that some other error occurred.

An important concept to understand about `scanf` (and many other C input functions) is that a C program sees input as a stream of characters, not as a sequence of lines of text. `scanf` will consume just enough characters to do its job, or to find out that it can't do its job. Remaining characters on the input stream are not consumed--instead they remain in the stream, waiting for the next input operation.

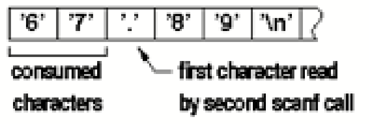
What To Do - Part I;

Copy the file `lab2exe_F1.c` from D2L. Read the program. Compile it, then run it several times, first with sensible input, then with various forms of bad input.

For example observe what happens when you enter letters in response to the prompt for an integer? What if you type two integers on the same line? What if you type 67.89? What if you type one integer followed by letters? (You don't have to write down the answers.)

An Explanation:

The following picture shows what happened when you typed in 67.89 in response to the prompt for an integer. The area of the memory that stores the stream of character inputs is called **'buffer'**.



The picture shows the stream of input characters after the first call to `scanf` was complete. Here is what the first call did:

- Read the '6', saw that it was a digit and could be used as part of an int.
- Read the '7', saw that it was a digit and could be used as part of an int.
- Read the '.', saw that it was not a digit and could be not be used as part of an int. The '.' was put back on the input stream so that it could be read by the next input operation.

This is an important exercise. Don't hesitate to ask instructor or a TA for more information. You will not be able to complete the next exercise if you do not fully understand this exercise.

What to Submit:

There is nothing to hand in for this exercise (part I).

What to Do – Part II:

Copy the file `lab1exe_F2.c` from D2L. Read the program. Compile it, then run it for 6 times, enter the inputs as listed in the following table, and complete the table for the values of n, i, and d.

Run #	Your inputs	What is the value of n	What is the value of i	What is the value d
1	`12      0.56`			
2	`5.12    9.56`			
3	`12      ab`			

4	ab 12			
5	5ab 9.56			
6	13 67			

What to Submit for part II:

This part will be marked. You should submit your completed table as part of your lab report.