ENSF 619 – Fall 2020
Lab 1 – Tuesday Sept 18

Department of Electrical & Computer Engineering
University of Calgary

M. Moussavi, PhD, PEng

Important Notes:

1. in this lab you can work with a partner (groups of three or more **are NOT allow**). 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 the popular method of program development called, **pair-programming.** This method which is normally associated with "Agile Software Development" techniques. In this method two programmers work together normally on the same workstation (you may consider a zoom session for this pupose). While one partner, the driver, writes the code, the other partner, acts a an 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, please submit only one lab report with both names. Submitting two lab reports with the same content will be considered as copies and plagiarism.

2. When submitting your source code (C or C++ code, files with the extensions: .c, .cpp, or .h), make sure the following information appears at the top of your file:
   a. File Name
   b. Assignment and exercise number
   c. Lab section
   d. Your name
   e. Submission Date:

   Here is an example:

```
/*
 *  File Name: lab1exe_F.c
 *  Assignment: Lab 1 Exercise F
 *  Completed by: Your Name (or your team name for group exercises)
 *  Submission Date: Sept 12, 2020
 */
```

Objectives:

This lab assignments is designed to give you some experience with the process of developing, compiling, running a simple C program, and learning basic constructs of C programming such as:

- Using standard input/output library functions printf and scanf.
- Using C/C++ control structures and more importantly pointers.

Please heed these advices:

1. Some exercises in this lab and future labs will not be marked. **Please do not skip them**, as unmarked exercises are as important as other exercises.

2. Some students skip directly to the exercises that involve writing code, skipping the sections such as "**Read This First**", or postponing the diagram-drawing until later. That's a bad idea for several reasons:

f. "**Read This First**" sections normally explain some of technical or syntax details that may help you to solve the problem, or may provide you with some hints.
g. 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:

Your lab reports must be submitted electronically on the D2L Dropbox before midnight (11:59 PM) on Tuesday Sept 22. All of your work should be in a single PDF.

- For instructions about how to provide your lab reports, study the posted document on the D2L called: `How to hand in your lab assignment.`

Important Notes:

- Some lab exercises may ask you to draw a diagram, and most of the students prefer to hand-draw them. In these cases, you need to **scan** your diagram with a scanner or an appropriate device such as your mobile phone and insert the scanned picture of your diagram into your PDF file. A possible mobile app to scan your documents is **Microsoft Lens** that you can install it on you mobile device for free. Please make sure your diagram is clear and readable, otherwise you may either lose marks, or it could be impossible for TAs to mark it at all.

Marking Scheme:

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

Exercise	Marks
A	no marks
B	8 marks
C	no marks
D	marks
E	4 marks
Total:	**17 Marks**

Exercise A: Creating a C source file, building and running an executable

Read This First:

If this is your first attempt to develop a C program, please do it so that you start to become comfortable with program development in C and particularly using your favorite development environment such as Visual Studio, Xcode, Geany, or smple editor such as Notepad++.
Before starting this exercise, please read the slides on "**Getting Started**".

What to Do:

- Start up favorite editor or IDE.
- Into the editing area, type exactly in all the following C code:

```
#include <stdio.h>

int main(void)
{
  int a = 0, b = 0;
  printf("Please enter a value for variable a:\n");
  scanf("%d", &a);
  printf("Please enter a value for variable b:\n");
  scanf("%d", &b);
  printf("The values of a and b are %d for a and %d for b.\n", a, b);
  printf("The value of a %% b is %d.\n", a % b);
  return 0;
}
```

- Now save your file as: `lab1exe_A.c`
- If you are using a text editor and Cygwin, or light IDEs such as Geany, on your operating system terminal navigate to your working directory (the same directory that you saved your `lab1exe_A.c`, then on the command line enter:

  ```
  gcc -Wall lab1exe_A.c
  ```

If the command succeeds, an executable file called `a.exe` (or on a Mac computer, `a.out`) will have been created. If the command fails -- which will be indicated by one or more error Messages--go back to your editor and fix the code, save the file again, try `gcc` again.

- Once you have an executable, run it a few times by using the command

  ```
  ./a.exe (or ./a.out on a Mac machine)
  ```

over and over.

Try different inputs each time; see what happens if you enter letters or punctuation instead of numbers.

Hint: You don't need to type '`./a.exe`' over and over! You can use the up arrow on your keyboard to retrieve previously entered commands.

Of course, if you are using more advanced IDEs such as Visual studio, you should use the features of those tools and you don't need to enter `gcc` command or run the executables such as `a.exe` and `a.out`.

Exercise B - Projectile Time and Distance Calculator

Read This First - A Brief Note on scanf Library Function

`scanf` is a Standard C Library function that can be used to read terminal input into variables. The first argument to `scanf` is a string constant with one or more format specifiers, like `%d` or `%lf`, and the remaining arguments are the addresses of the variables into which `scanf` will put the input data. Here is the list of format specifiers for **some** of the simple C data types:

C Data type	Format Specifier
`int`	`%d`
`float`	`%f`
`double`	`%lf`
`char`	`%c`
`long int`	`%ld`

For example, if **x** is a double data type, you can use the following syntax to put the input into **x**:

```
int n;
double x;
printf("Please enter the value of x: ");
n = scanf("%lf", &x);
```

In this example, when `scanf` successfully reads a value for `x`, it returns 1. But In general, `scanf` returns the number of items that has read successfully. And if it fails to read any number because of invalid user's input, returns zero. For example, if user instead of a number enters a few letters such `'abcd'` for x, `scanf` fails and the value of n will be zero. Also, in the following example if user enter two integer numbers for x and y, the value of n will 2.

```
int x, y, n;
printf("Please enter the value of x: ");
n = scanf("%d%d", &x, &y);
```

Read This Second

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:

$$d = \frac{v^2}{g}\sin(2\theta)$$

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 reaches the maximum horizontal distace the followig formaula can be used :

$$t = \frac{2v\sin\theta}{g}$$

In this exercise you will complete a given C source file called `lab1exe_B.c` that prompt 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 `lab1exe_B.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 missing functions as follows:

Function create_table: which is called by the main function, receives the projectile initial velocity and displays a table of 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 (ν) and returns projectile travel time (t).

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

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

Notes:

- To use C library function `sin`, you need to include header file `math.h`. And, if you are compiling and running the program from command line use `–lm` option to link the math library:

  ```
  gcc -Wall –lm lab1exe_C.c
  ```

- Please pay attention for constant values of π, and gravitation acceleration, g, the following lines are already included in the given file. using preprocess or #define is another way of indicating constants in C:

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

  As an example anywhere in the code that compiler finds the word `PI` it will be replaced with `3.141592654.`

What to Submit:

Submit the copy of your program (your code and the program output) as part of your lab report in PDF format. You don't need to upload your actual source code (the .c file). Only it must be copied and pasted into your lab report along with the program's output.

Exercise C – Introduction to Pointers

This exercise will not be marked, and you shouldn't submit anything.

Read This First

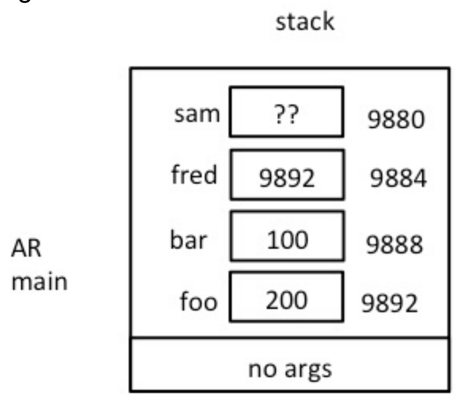
This is an important exercise in ENSF 619. If you don't become comfortable with pointers, you will not be able to work with C. Spend as much time on this exercise as is necessary to understand exactly what is happening at every step in the given program.

What to do

Download the file `lab1exe_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 **<u>use the given address numbers as values of the pointers (don't use arrow notation in this exercise)</u>**. To understand how to draw these diagrams the solution for point one is given in the following figure:



After you completed the exercise, compile lab1exe_C.c and check your predicted output against the actual program output.

Exercise D: Pointers as function arguments

What to do – Part I

First copy the file `lab1exe_D1.c` from D2L. Carefully make diagrams for point one in this programs using "**Arrow Notation**" as is discussed in the set of slides called `03_Activation Records` on the D2L, then compare your solution with the posted solution on the D2L.

Exercise D: Pointers as function arguments

What to do – Part I

First copy the file `lab1exe_D1.c` from D2L. Carefully make diagrams for point one in this programs using "**Arrow Notation**" as we discussed during lectures (you don't need to use made-up addresses, the way that we did it in the previous exercise). Then compare your solution with the posted solution on the D2L.

There is nothing to submit for this part

What to do – Part II

Now download the file `lab1exe_D2.c` from D2L and draw AR diagram for point one in this file.

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

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

Read This First

Here is an important note about terminology:

- *Don't* say, ``Pointers can be used to make a function return more than one value.'' A function can never have more than one return value. A return value is transmitted by a return statement, not by a pointer.
- *Do* say, ``Pointer arguments can be used to simulate call by reference,'' or, ``Functions with pointer arguments can have the side effect of modifying the variables that the pointer arguments point to.''

What to do

Make a copy of the file `lab1exe_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.