The Lab 1 task will help us understand core components of programming in C.

References:
https://cplusplus.com/reference/cstdio/printf/
https://cplusplus.com/reference/cstdio/scanf/

The Learning Outcomes for this Lab are:

- Datatypes in C
- Variables
- Branching - if statements, multiple ifs, nested if, switch statements.
- Pointers
- Arrays in C
- Looping/ Repetition
- Functions in C
- C Structs

1) Data Types, Variables and Branching, C Strings

- Formatted string literals (using format specifiers for printing and reading from the standard input
  - Integer (%d or %i)
  - Floats (Single precision floating point - %f)
  - Double (Double precision floating point - %lf)
  - _Bool (%d or % i) - requires 0 or 1 so its an integer type
  - Char (%c) - This is also integer type , use atoi to convert
  - C Strings - (%s) - note C strings are characters arrays which are

Task 1:
Professor Mwaura (He/Him/His) has contracted you to create a program that stores details about his students. Below are details that he would like the program to take and print them out.

1) Student Last Name
2) Student Age
3) Student GPA
4) Student Final Exam Score
5) Student Grade - Note that for the student Grade, Professor would like your program to determine the Grade based on the following:
   a) 90 and above : A
   b) 85 and below 90 : B
   c) 80 to Below 85: C
   d) 75 and below 80: D
   e) Anything below 75: S

Instructions: This task is to be completed in the main function. Your program should allow for the user to enter this data via standard input and output the data via a standard output.

Checkpoint 1 (10 points) : Show your implementation and execute your program for the Teaching Assistant or Professor.

2) Pointers in C

The definition of a variable is: “a named location in memory”. This means a variable is not the value that we are storing; it is what you might think of as a physical addressing with a name.

For instance: I might say, I am at 43 degrees N, 70 degrees west; If I said so and you wanted to come visit me, you would need to figure out a system to read degrees using a compass. However, if I said I am in Portland, ME then it is possibly easy for you to remember exactly where I am. Note the longitude and latitude that I gave you are for Portland, ME.

In the same way, we use a variable identifier (i.e. name of variable) to refer to the address in memory that we are storing a particular value. How can we know the address? We can use an “address of” operator to know the address:

E.g.

```
int age = 29; // variable declaration and initialization
printf("My age is %d and its stored in %p \n", age, &age)
```

Note: the format specifier for printing out the address is %p - to show that this is a pointer (or a reference to the address) . Recall also the address is really a number, so %p will print this number as a hexadecimal number. We could also use %lu - a long unsigned integer, though it will throw a warning.

If we can print out this address, it also means we can store it. How do we store it: Well, recall that the way to store things is to use a variable. So we can create a variable to store an address but it then must be a special variable, right? We use a “reference operator” which is “ *” to do the declaration, for instance

```
//declaring a pointer variable; recall using RLWM, we read this as pAge is a pointer to an integer
int * pAge;
```

Knowing the address of something is cool, because we can use it if we want to change the value stored there or even look at exactly what is stored there. This means we can also store the address

```
//Now we can use pAge to store address of an integer such as age

pAge = &age; //& is address of operator or reference operate
```

```
//if we want to print an address associated with age, we can just print pAge.
//We can also change the value at the address by using a dereference operator

*pAge = 31;

printf("My age is %d and its stored in %p \n", *pAge, pAge);
```

Task 2: Professor Mwaura would like to know the addresses why particular attributes of the students are kept. For this Task, you can use the address of the operator to print out the addresses.

Instructions: Modify your code to either directly print out the addresses of the variables used (using the 'address of ' operator) or alternatively use pointer variables to print out the addresses.

Answer the following questions:

- How many bytes are used to store the following data elements:
    - Student Last Name
    - Student Age
    - Student GPA
    - Student Final Exam Score
    - Student Grade
- What is the beginning and the last byte address of the Student Last Name variable? What does this tell you about the storage mechanism for this element?

Instructions: This task is to be completed in the main function. Your program should allow for the user to enter this data via standard input and output the data via a standard output.

Checkpoint 2 (20 points) : Show your implementation and execute your program for the Teaching Assistant or Professor. Also show and record your answers to the above questions and show them to your instructor (TA/Prof).

Functions

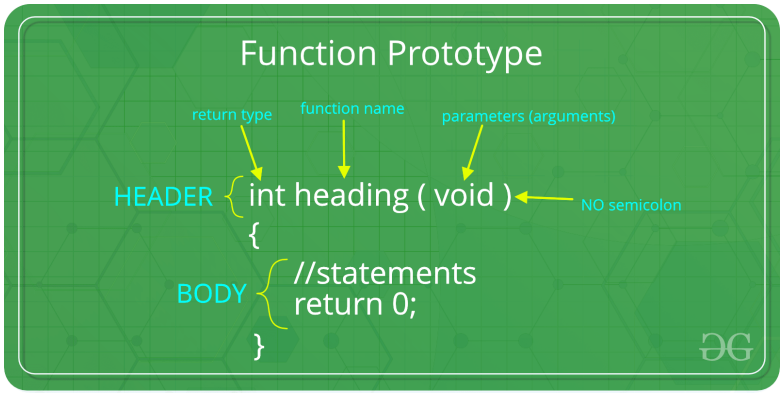
If you look at your current programs, everything from data collection to processing is happening in the Main function. In this part of your work, you want to collect the data separately. For instance you would want to collect the names of students on a separate function,

How does C deal with functions; let's go back to the main function.

```
int main (void) {..
…..
…..
return 0;
```

}
What does it mean - recall the RLWM -

It turns out that C functions have a return type, the function identifier (its name) and the arguments it is taking (also called formal parameters) as well as the body of the function.



Void is a type that we use when a function is not receiving any argument or returning anything. Function could return any type of data and they can also return addresses (i.e. we can return pointers).

Instructions: In this exercise we are moving away from carrying all our computation in the main function. Instead we are going to create Functions where all processing is carried out and we are going to access those functions from the main functions. This turns our main function as the **DRIVER**.

Good programming in C requires that you create Function signatures (or function headers) before the main function and then create Function implementation below the main function.

Create the following functions

1) Four functions that **get(or allow to input)** the required data items. The details of these functions are as follows:
    1) Student Last Name (20 points); The functions should allow for a user to enter the student last name. This function should NOT return anything to the main function. The function should have, as its argument, the array variable that was declared in main. Essentially, the function will populate the same variable declared in main without returning anything to main,

        ```
        void yourFunctionName(char name[]); //function header
        ```

        Note: Think deeply about how you would go about fulfilling this.You can try many different variations of it.

        Answer the following:

- How did you implement this function?
- In class we mention that the name of an array function holds the address of the first element: Try to print out the name of the lastname array variable in main and also print out the name of the formal parameter (argument) in your function. How do they compare?
- What did you learn about CStrings and perhaps (arrays in general) with regard to the C language, after completing this function?

2) Student Age (5 points): This function should not receive anything and will return an integer

```
int yourFunctionName(void); //function header
```

3) Student GPA (5 points) - This function will not receive anything and will return the student GPA
4) Student Exam Score (5 points) - This function will not receive anything but will return the student exam score.

2) Letter Grade Function (5 points): This function receives a students exam score and determines what their letter grade is; The criteria for determining letter grade is given in Task 1.

3) Print data Function (10 points): This is a function that outputs the data onto the standard output. The data should include the student ‘s letter grade; although this function will not be receiving the letter grade as part of its arguments.
The function signature looks like:

```
return type  printData(last name, student age, student gpa, exam score)
```

Note: whereas it is not specifically mentioned, you may need to create a function that clears the buffer/spool.

Checkpoint 3 (see all points in different parts) : Show your implementation and execute your program for the Teaching Assistant or Professor. Also show and record your answers to the above questions and show them to your instructor (TA/Prof).

Submission (20 points): Your code will be submitted to your Github repository. If you have not been working on this lab using the Khoury Servers, your will need to do the following:

a) Push your work to your github repository from the local directory you have been working from.
b) Pull the code from to your directory in Khoury Server and test that your code works there.

c) Once satisfied, push this back to your github repository.
d) The TA will grade it up on the server and away last points.

All the best