Programming – TU856/1 & TU858/1

Lab 1 – Tuesday, October 4th, 2022

Note:

- Create a new file for each question.
- Please use file names of 10 characters or less for each program you write.
- Always save your C source code files with .c extension at the end of each filename.
- **Do not include blank spaces in file names** - use an underscore (_) or camelCase instead.

1. Enter the first program discussed in lecture class. This involved declaring 3 variables (int, float, char). Name this program **MyFirstIO.c** and save it in an appropriate folder on your laptop/desktop/cloud folder possibly called *week3*. This should be created within a parent folder called *C Programming*.

   Compile and run this program. Make small changes to your C code and re-compile and run the program. Notice the difference(s) your changes have made.

   Remember, use appropriate white space, i.e., blank lines, in your code. Do not squash all the code together, line after line !!

2. Copy the code below into a new file. Correct the errors, compile, and run it.

```
#include <stdio.h>;
main[]
{
   /*Program to illustrate errors in a C program.
   int num1, num2;
   float num3;

   num1 = 400;

   600 = num2;

    PRINTF("The value of num1 is %d" num1);
    PRINT(" num2 is %d");
]
```

3. Write a new program (i.e., create a new file) that will display to Standard Output (i.e., the screen), your name and address on separate lines. **Note**: you do not need to create variables to display a string. Just place the string you wish to display to Standard

Output inside the double-quotes of a printf() statement. We will learn later how to store and manipulate strings using a data structure in C.

4. Edit your program in Q3 to include an empty blank line between your name and each line in your address. Also display today's date following the address.

5. Write a new program (new file) that declares 3 variables (int, float and char). Assign any value of your choice to these variables. Using a *printf()* statement, print the contents of these variables using the **wrong** delimiter i.e. use a %d for a float, %c for a float, %f for a char. Compile and run your program. What happens?

   Change your program to use the correct delimiter, i.e. %d for an int, %f for a float, %c for a char. This time, use %d for a char. Notice anything different? If so, what is causing these differences? Think about what we talked about in lecture class.