# ELEC 334 - Homework #2


Reminders:

- Please read carefully, and answer accordingly.
- Submit your solutions in a PDF file and any additional files asked from you.

**Problem 0** [`0 pts`]. **Test setup.**
This is a C project setup to get you accustomed with project organization. Create a directory, get the files from https://gist.github.com/fcayci/61c60602ac98304741fe1728a80d1d01 and place them in the directory, then compile and run your code. Example Makefile is given if you need it.

**Problem 1** [`10 pts`]**. Pseudo-random number generator.**
Draw a **flowchart** for a pseudo-random number generator, then write its **C function**. The generated random number should be between **[1, 15]**. It should be a pure C function without any external function calls. (i.e `rand()`) Explain your method.

You should have `main.c`, `myrand.c` and `myrand.h` files.
`myrand.c` should have your random number generator function definition.
`myrand.h` should have a function declaration.
`main.c` should include your myrand function and call it.

**Example:**

```
./myrand
> 12
./myrand
> 3
```

**Problem 2** [`15 pts`]. **Test your random number generator.**
Draw a **flowchart** and write a **C function** that will test your random number generator. Compare it against `rand()` function from standard C library. For this, generate a lot of random numbers (>100k), keep each number's count, and print the **normalized results** at the end in tab separated order.

You should have `main.c`, `test_random.c`, `test_random.h`, `myrand.c` and `myrand.h` files.

**Example:** An example of generating 3000 random numbers between 0 and 2 is given below.

```
test_random
> results from myrand():
> 0: 0.334    1: 0.308    2: 0.358
> results from rand():
> 0: 0.315     1: 0.341   2: 0.344
```

To elaborate, from the numbers that are generated by `myrand` a total of `33.4%` of them are `0`, `30.8%` are `1`, and `35.8%` are `2`.

**Problem 3** [`15 pts`]**. Instruction Decode**
Using ARMv6-M Architecture Reference Manual write the hexadecimal representations for machine code for the following instructions, explain bitfields.

```
ldr  r5, [r6, #4]
mvns r4, r4
ands r5, r5, r4
adds r0, r0, r1
add  r0, r0, r1
subs r2, r4, #2
asrs r2, r4, #21
str  r5, [r6, r1]
bx lr
bne 0x12
```

**Problem 4** [`10 pts`]. **Instruction cycle times**
For the instructions given in Problem 3, find how many cycles each one takes.

**Problem 5** [`5 pts`]. **Assembly delay function**
Write a delay function in assembly that will wait for a given number.

**Problem 6** [`15 pts`]. **Assembly LED toggle**
Write assembly code that will toggle an LED connected to PortB pin12 at roughly 1 second intervals. Assume LED is connected as active-low, and processor speed is 16 Mhz.

**Problem 7** [`15 pts`]**. Assembly Hamming distance**
Draw a **flowchart** that will find the **hamming distance** between two values located in **mem[0x14224]** and **mem[0x14228]**, and write the result back to **mem[0x1422C]**. Then write the code in **assembly**.

**Problem 8** [`15 pts`]**. Assembly Average**
Write a program that will calculate the integer average of numbers located starting address labeled as `myarray` below up until there is a 0 in the array. Write the average value at memory address 0x20000000. Draw the **flowchart**, and write **assembly** code.

```
myarray:
  0x20000100: 12
  0x20000101: 27
  0x20000102: 30
  0x20000103: 29
  0x20000104:  8
  ...
```