University of Nottingham
UK | CHINA | MALAYSIA

EEEE1042 - **Practical 5**
Strings functions, random numbers, 2D arrays.

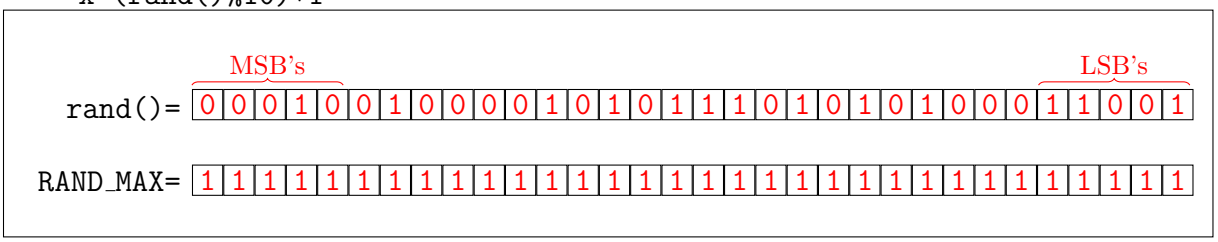
1 Generating Random Numbers

In the notes of Lecture 5, you were shown how to generate random numbers using the `srand()` function to seed the random number generator and the `rand()` function to generate a random integer between 0 and `RAND_MAX` inclusive. In order to use `rand()` you need to `#include<stdlib.h>` in the header together with `#include<stdio.h>`. One of the messages from the lecture was that to generate a random sequence between 1 and 10, you should use a formula that uses the MSBs like this:

```
x=1+(int) (10.0*rand()/(RAND_MAX+1.0));
```

rather than the LSBs like this:

```
x=(rand()%10)+1
```



In this exercise we are going to see if we can detect any difference between using the LSB's and MSB's to generate random numbers.

1. Write a C-program that generates `N` random numbers using each of the 2 above methods above of calling `rand()` (where you `#define N` in the header) and writes them to a file.

2. For the method that uses the MSB above, write the output of the `N` random numbers into a file called `"MSB"`. When complete, the file `MSB` should have a single column of `N` random numbers between 1 and 10 generated using the above MSB formula.

3. While for the method that uses the LSB above, write the output of the `N` random numbers into a file called `"LSB"`. When complete, the file `LSB` should have a single column of `N` random numbers between 1 and 10 generated using the above LSB formula.

   Save your program to the file: `yourName_practical5_EEEE1042.1.c` and submit it to Moodle. Be sure to have properly commented your code.

4. View the files `MSB` and `LSB` to check that you do indeed have a single column of random numbers in each case. From the command terminal you can use the `cat` command: `cat filename` to display the contents of your respective file. Or you can open your file in emacs with `C-x C-f filename`. Use `C-g` if you want to exit without typing a filename. In general `C-g` backs out of any half-completed commands. `C-x C-k` kills the buffer after you've confirmed the contents look correct (and allows emacs frees up the memory for that file).

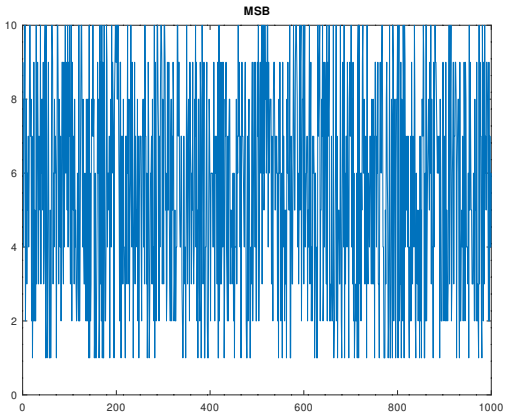
5. Here is an Octave script called `plota()` to plot the contents of your `MSB` and `LSB` files:

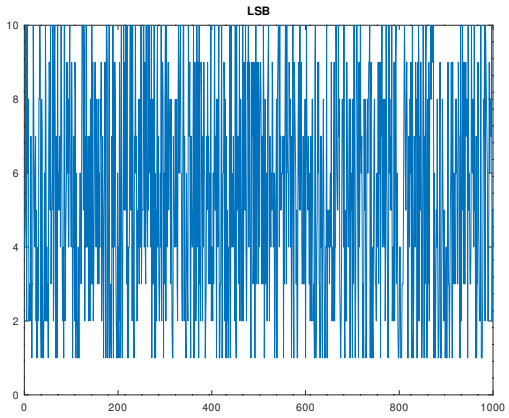
```
#!/usr/bin/octave
function plota(filename)
  a=load(filename);
  plot(a);
  title(filename)
  print('-deps2', '-color',[filename '.eps']);
endfunction
```

Type in and save the above script to a file in your current working directory called `plota.m`. In emacs, you would type `C-x C-f` to find a file then type `plota.m` as the name of the file to find. It will open `plota.m` as a new file to type in your program. The `.m` extension indicates its a Matlab or Octave script. Please remember when you are copy-and-pasting that the single quote character ' and underscore character _ do not copy-and-paste over into C-programs properly, you will need to retype these by hand. You can then execute this script with:

```
octave --eval "plota('MSB')"
```

This command uses octave to run the `plota()` script which will plot the contents of the `MSB` or `LSB` file to a corresponding file called `MSB.eps` or `LSB.eps`. You may either run the command from the terminal, or put it within your make file and press F5 as usual (be sure to comment out the other parts of your make file that you don't want to run). Either way after running the script you should end up with 2 plots `MSB.eps` and `LSB.eps` that look similar to the following:





which were created using `N=1000`. Type `la` to see the contents of your directory and

make sure the two new files are there. Type `evince LSB.eps` to view the `.eps` file you just generated, with the `evince` viewer.

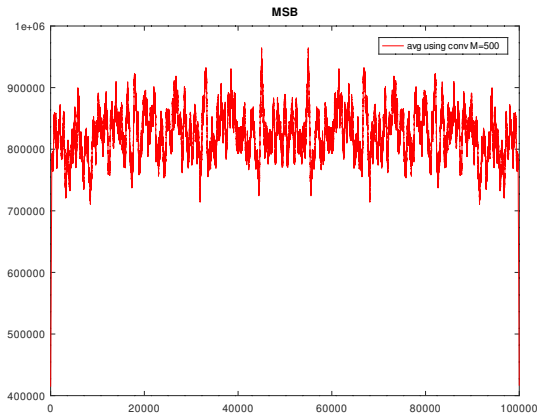
6. To detect the "randomness" of these two sequences, we are going to look at an approximation of the **power-spectral density** (PSD) of these two random signals you have generated. If there is any correlation, it could show up looking like coloured noise in the PSD. If the numbers are uncorrelated, it should have a "white" spectrum. Here is a second program also written in Octave that will calculate and plot a smoothed version of the `fft` of your signal:

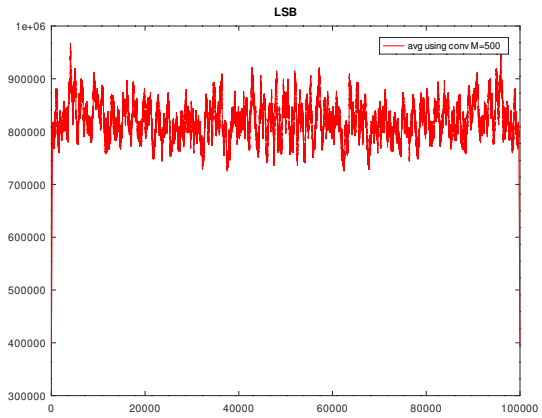
```
#!/usr/bin/octave
% plot the abs fft of signal in filename
% N is an averaging number to average the FFT over to reduce the noise
function plotf(filename,M)
  a=load(filename);
  a=a-mean(a); % subtract the mean
  a=fft(a); % take the fft of the data squared
  a=abs(a.*a); % take the fft of the data squared
  b=conv(a,ones(M,1)/M); % Calculate moving average
  b=b(M/2:end-M/2); % Remove the edges
  plot(b,['r;avg over ' num2str(M) ' ;']); hold off
  title(filename)
  print('-deps2', '-color',[filename '_fft.eps']);
endfunction
```

Type in and save the above script to a file in your current working directory called `plotf.m` similar to what you did for `plota.m` above. Because the signal is very noisy, `plotf()` averages together `M` consecutive samples of the PSD where `M` is an input to the function. You should therefore call your function passing in two inputs:

```
octave --eval "plotf('MSB',M)"
```

where you need to choose an appropriate value for `M` for the number of samples averaged together. If you chose `N=1000`, you might try `M=10`. Using parameters `N=100000` and `M=500` we get plots like these:





Generate plots similar to these on your side and for your own chosen values of `N` and `M` and upload them to Moodle. A white (flat) spectrum would tend to indicate no correlation in the signal.

2 Converting between Farenheit and Celcius

Write a program that inputs 6 numbers representing temperatures in degrees farenheit from the user through the scanf function storing them in an array. The program then converts these numbers into celcius using the formula $C = \frac{5.0}{9.0}(F - 32)$ and prints out a table of the conversion.

After converting from farenheit to celcius, have your program do the inverse by converting celcius values back to farenheit using the formula $F = \frac{9.0}{5.0}C + 32$.

To store your data, declare two float arrays:

```
float farenheit[6];
float celcius[6];
```

Use the POP modular programming by writing functions that handle each task of:

1. Inputting 6 numbers from the user into the `*farenheit` array
2. Computing the `*celcius` array from the `*farenheit` array
3. Printing the table with original farenheit temperatures on the left and the converted celcius temperatures on the right.

==========

Now re-use (most) of the above functions to handle the inverse tasks of:

4. Inputting 6 numbers from the user into the `*celcius` array
5. Computing the `*farenheit` array from the `*celcius` array
6. Printing the table with original celcius temperatures on the left and the converted farenheit temperatures on the right.

The output of your program should resemble the following:

```
$ ./a.out
Please enter farenheit value 0
0
Please enter farenheit value 1
20
Please enter farenheit value 2
40
Please enter farenheit value 3
60
Please enter farenheit value 4
80
Please enter farenheit value 5
100
```

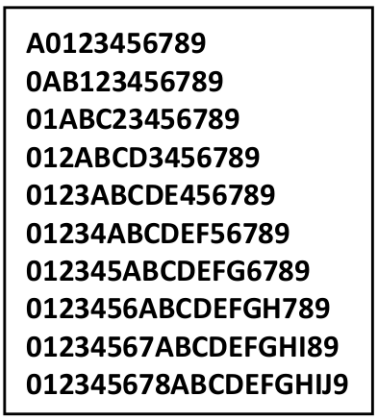
```
 Farenheit | Celcius
---------------------------
0.00      | -17.78
20.00     | -6.67
40.00     | 4.44
60.00     | 15.56
80.00     | 26.67
100.00    | 37.78
Please enter celcius value 0
0
Please enter celcius value 1
20
Please enter celcius value 2
40
Please enter celcius value 3
60
Please enter celcius value 4
80
Please enter celcius value 5
100
Celcius | Farenheit
---------------------------
0.00    | 32.00
20.00   | 68.00
40.00   | 104.00
60.00   | 140.00
80.00   | 176.00
100.00  | 212.00
```

Save your program to the file: yourName_practical5_EEEE1042.2.cpp and submit it to Moodle. Be sure to have properly commented your code.

3 C String manipulation

Complete the missing part from the following code to get the output shown in the Figure below.

```
#include <stdio.h>
#include <string.h>
int main() {
  char str1[] = "0123456789", str2[] = "ABCDEFGHIJ", str3[100];
// Complete your code below here using str1, str2 and str3
// using C string-functions like strcpy, strcat, strlen etc...
}
```



Use a `for` loop to iterate over the length of one of the strings. Write a function that creates and then prints the combined string as shown in the above Figure for each iteration through the `for` loop. Use the functions `strcpy` and `strcat` and `strlen` that were shown to you in the Lectures to achieve your target. Save your program to the file: `yourName_practical5_EEEE1042.3.cpp` and submit it to Moodle. Be sure to have properly commented your code.