BIOINFORMATICS. AN INTRODUCTION TO MATLAB PROGRAMMING

The course will take place in five classes. Every day a sheet of exercises will be given to be solved in the class. In each sheet there will be some mandatory exercises that must be sent to jklett@cbm.uam.es and acortes@cbm.uam.es. These problems will be explained during the classes, and there will be enough time to complete them. Students are strongly encouraged to do all problems.

Please, before sending the exercises, note:

- Programs should be sent by e-mail in separate .m files with the original exercise name.
- Student name should be the first line of the program as a comment. Comments in the programs and text messages may be in English or Spanish.

Once the exercises are corrected, a second deadline will be available for fixes. These exercises will be evaluated as passed/non-passed. All exercises must be passed in order to continue with the bioinformatics course.

At the end of the classes period, all student must perform a project related with the theoretical part of the bioinformatics course. Details on project evaluation will be discussed with the project list.

Day 3 problems

1. **Suma.** Same as suma_script, but in functional form: Write a function that takes a vector as input, finds the sum of its components, and gives it as output. You are not allowed to use Matlab's function "sum", which is equivalent.

2. **Encontrar.** Same as Encontrar_script, but in functional form: Write a function that requires two input arguments: a vector and a number. The output must be a vector which contains the indices of the elements of the input vector that are equal to the input number. If there are no coincidences, the output should be an empty vector. This is identical to the "find" Matlab funtion. Do not use it!

3. **Encontrar_eficiente.** Same as Encontrar, but must use preallocated variables.

4. **Maximo.** Write a function that takes a vector as input and gives two outputs: The maximum of the vector, and the index corresponding to it. If there are more than one element equal to the maximum, the index must correspond to the first one. This is equivalent to the "max" Matlab function. You are not allowed to use it.

5. **Ordernar (mandatory).** This function will have one input argument, which will be a vector. It will have two output arguments, which will be two vectors of the same length as the input one. The first output vector must be the input vector, but with its elements sorted in increasing order. The second output vector must contain the positions occupied by each element of the sorted vector in the original one. This function is equivalent to Matlab's function "sort". As you probably imagine by now, you cannot use "sort" for this exercise. You may use functions "min" and "max".

6. **Bisiesto.** Write a script that takes as input a year and has no output arguments. It must display a message showing if it is a leap year or it is not. The leap year rules are the following: A year is leap if it is divisible over 4, except those divisible over 100. But also, years divisible over 400 are leap. For example, 1996, 2000 and 2004 were leap years. On the contrary, 1900 was not, because it is divisible over 100 and not divisible over 400. In order to know if *a* is divisible by *b*, the remainder of *a*/*b* must be 0. For Matlab, *a* is divisible over *b* if *rem(a,b)==0*.

7. **Bisiesto_comprimido.** Identical to the previous one ("Bisiesto"), but with only one "if" statement (you have to use logical operators "&", "|").

8. **Histograma (mandatory).** Write a function with two input arguments: A vector with the data, and a scalar with the number of bins. It must generate the histogram of these data, plotting it as a bar graph (use instruction "bar"). Also, it must have two output vectors. The first one must contain the number of data falling in each bin, and the second one must contain the centers of the bins. Bins must be equispaced, and must contain the full range of data (that is, the lower extreme of the first bin must match the minimum datapoint, and the higher extreme of the last bin must match the maximum datapoint). Each bin must contain data that are higher than its lower extreme, and lower or equal than its higher extreme (except for the first bin, which must also contain data equal to its lower extreme). This function is similar to Matlab's function hist, and you cannot use hist for this exercise. You may create the data vector with instruction randn (thus, the histogram will be a nice gaussian).