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 2 problems

1. **PesaditoWhile.** Modify the "Pesadito" script so that it gives the same output but using a "while" loop.
2. **CuentaTiradas.** Write a script that simulates the flipping of a coin, until it gets three heads. It must show the numbers of flippings that were needed.
3. **Millonario.** A method to become a millionaire is to go to a Casino and bet in the roulette. You bet 1€ to red (that has roughly 50% probability). If you win, you get 2€. If you lose, then you bet 2€. Again, if you lose you bet 4€, and so on. In this way, when you finally win (and this must happen at some point) you recover all your previous bets, plus 1€. Then you start again. Write a script that simulates this method, telling you in the end how many bets you needed to win, and how many euros you betted the last time.
4. **Arruinado.** Modify the previous script, so that the casino has a money limit of 1000€, so that if your bet exceeds 1000€ you must stop playing, losing all the money you have betted. The program must say if you win or if you lose. Run it several times, to find out that you will not become a millionaire after all.
5. **ArruinadoHistorial (mandatory).** Now the casino has no limit, but you have only 1000€ at the beginning. After each bet, update the money that you have left. Store the results in a vector (one element for each bet). One example in which you lose 3 times and then you win would be [1000 999 997 993 1001 …]. The program finishes when you run out of money. At the end, plot the vector to see the history of your misery.

6. **The Matrix.** Load the file "matrix.mat" into the workspace, which will give you the matrix M. First, determine the size of M, i.e., the number of columns and rows. In the command window, calculate the mean and standard deviation for each column of M, as well as the maximum and minimum of each column. Change the matrix by setting some elements to Inf and NaN. Again calculate the mean of the columns.
7. **The Matrix II: Correlation.** Reload the matrix. Plot each column of the matrix into the same figure. Having a close look at the plot, you can see that two of the columns might be correlated. In order to check a possible correlation, make two plots, in each using the first column of M as x-values and the 2nd or 3rd one as y-values. Use unconnected dots to represent the data. Which column is correlated to the first one?
8. **Tabla5.** Write a script that creates a matrix with the results of multiplying all the numbers from 1 to 5 between them ("all against all").
9. **Tabla5_nopairs.** Create a script similar to the previous one ("Tabla5") but that avoids printing the same pairs twice (1*5 = ~~5*1~~). Hint: try with "i=j:N"…
10. **Sinxplot.** Write a script that represents graphically the following functions: sin(x), sin(2x), sin(3x), sin(4x) for $0 < x < 4\pi$. The graph should show the values with 0.1 intervals. Remember using loops and vectors!
11. **Suma_script.** Write a script that finds the sum of the components of a vector (which must be previously defined). You are not allowed to use Matlab's function "sum", which does the same thing as this script.
12. **Acumulada.** Write a script that finds the "cumulative vector" of a previously defined vector. Each element of the cumulative vector stores the sum of all previous elements of the original vector. For example, if the original vector is [1 1 3 2 4] the cumulative vector is [1 2 5 7 11]. This script does the same thing as Matlab's function "cumsum". You cannot use this function for the script.
13. **Encontrar_script (mandatory).** Write a script that requires two variables to be previously defined: a vector and a number. The program must find the elements of the vector that are equal to the number, and store their indices in another vector. If there are no coincidences, the created vector should be empty. You are not allowed to use Matlab's function "find".
14. **ProtAlign (mandatory).** Write a script that represents graphically the identities in sequence with itself for the following protein:

    'AAAEYDSLEYDSLGYENEAAAEYDSLEYDSLGYENE'

    The output should be a binary matrix (1=same amino acid; 0 otherwise). Use for the output the "imagesc" function.