1. Choose 3 variables from our surveys:

one Qualitative

one Quantitative discrete

one Quantitative continuous (use class intervals in this case, obviously)

create the most efficient algorithms to compute the frequency (absolute/relative/percentage) distribution of:

the 3 variables

the joint distribution of 2 variables (use a general “logic”, where variables could be any number, k=2.3,…).

Double check/compare the results using some DBMS functionalities you prefer (eg., access, oracle online, postgres, …) wherever possible.

To develop this algorithm I chose the Survey Personal Life and I converted the file in .tsv format, where every variable is separated by the tab \t. To allow the computation of the distribution, I’ve created a button that when it’s clicked triggers the compute\_freq function which triggers all the functions to calculate the requested distributions.

The first step to compute the distribution is parsing the .tsv file using a function called parser, which takes the path of the file used for our purposes and parses it to create a matrix parsed\_s, which will be useful for computing the distributions.

The second step was choosing the variables. I chose:

Expected work sectors as Qualitative

Hard Worker (0-5) as Qualitative Discrete

weight as Quantitative Continuous

After, I created three functions to calculate the different asked distributions:

discrete\_freq to compute the Discrete Distribution

continuous\_freq to compute the Discrete Continous Distribution

joint\_freq to compute the Joint Distribution

All this functions calculate also the relative frequency and the percentage frequency, by calculating the total times a value appears and the dividing it for all the frequencies previously calculated.

Moreover, for the joint distribution I chose two discrete variables, Expected work sectors and Hard Worker (0-5), specifing them in the written code.

2. For the following most important data structures (or others that you may want to suggest) recall how to:

loop (break/continue)

add/remove/get/set/check the existence of key/value

data structures: array, list, dictionary, sorted list, hashset, sortedset, queue, stack, linkedlist (or any other structure you think to be useful)

Note in a very concise way your finding also in your Js Cheatsheet, and, in case a corresponding Js object does not exists, create a simple equivalent class with all necessary corresponding methods to use it, similarly to c# (or any vb.net if you prefer).

To see how the functions for the requested data structures have been implemented, I suggest to follow the Links for the C# and JavaScript CheetSheets:

C# CheetSheet

JavaScript CheetSheet

3. Generate N uniform random variates in [0,1) and determine the distribution into class intervals [i/k, (i+1)/k), i = 0,…, k-1.

Play with N and k values and draw some conclusion on the “shape” of the distribution.

To compute this function I used two different techinques to choose the number of the variables n and the size of the interval k:

In the C# code, I generated this two variables using the function Math.random() which generates random number in a defined interval limited by 1000 for n and 10 for k.

In the JavaScript code, I generated this two variables taking them from the input inserted by the user.

Then, I’ve calculated the frequency of each number that is, how many random numbers are contained in each interval.

As we can see using different values of n and k, the “shape” of the distribution is uniform. This means that:

When n is sufficiently large and k is chosen appropriately, the shape of the distribution should closely resemble a uniform distribution;

In a uniform distribution, each interval is expected to have approximately the same number of values.

Moreover, the choice of n and k affects, clearly, the “shape” of the distribution since:

If we incrase n, the distribution becomes more uniform, because with a larger number of samples, the random values are more evenly spread across the entire [0, 1) range;

If we increase k, we can have a more detailed view of the distribution, but every interval will contain few values.