You must implement a Matlab function or a Python executable file called k\_means. Your function should be invoked as follows:

k\_means(<data\_file>, <K>, <initialization>)

If you use Python, just convert the Matlab function arguments shown above to command-line arguments. The arguments provide to the function the following information:

The arguments provide the following information:

* The first argument, <data\_file>, is the path name of a file where the data is stored. The path name can specify any file stored on the local computer.
* The second argument, <K>, specifies the number of clusters.
* The third argument, <initialization> specifiers how the initial assignment of data points to clusters is done. This argument can have two possible values: random, or round\_robin.

Example 1D and 2D datasets that you can test your code with can be found in the [toy\_data](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment6/toy_data) directory. In each file, each row corresponds to a data point, which can be a single number or a 2D vector. The two values of each 2D vector are separated by space.

Your code should work with any other files that follow the same format. You can assume that the input data will be either 1D or 2D, you do not need to worry about higher dimensions.

**Implementation Guidelines**

* To make the result deterministic, if the third command-line argument is round\_robin, the initial cluster assignments should be done in round-robin fashion. More specifically, the first object gets assigned to cluster 1, the second object gets assigned to 2, ..., the k-th object is assigned to cluster K, and so on. For example, for file [set1a.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment6/toy_data/set1a.txt), and K=3, this is how the initial assignments of points to clusters should be:
  + 7 --> cluster 1
  + 29 --> cluster 2
  + 11 --> cluster 3
  + 2 --> cluster 1
  + 16 --> cluster 2
  + 4 --> cluster 3
  + 37 --> cluster 1
  + 22 --> cluster 2

Note that cluster IDs range from 1 to K, and there is NO cluster ID 0.

* Your algorithm should terminate when the assignment of objects to clusters stops changing. In other words, the algorithm should stop if the assignment of objects to clusters at the end of the current iteration is identical to the assignment at the end of the previous iteration.

**Output**

At the end, your program should print the final cluster assignments. If the dataset is one-dimensional (like file [set1a.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment6/toy_data/set1a.txt)), the output should contain one line per data point, printed with the following formatting specification:

fprintf('%10.4f --> cluster %d\n', data\_point, cluster\_id);

If the dataset is two-dimensional (like file [set2a.txt](https://athitsos.utasites.cloud/courses/cse4309_fall2021/assignments/assignment6/toy_data/set2a.txt)), the output should contain again one line per data point, printed with the following formatting specification:

fprintf('(%10.4f, %10.4f) --> cluster %d\n', data\_point[0], data\_point[1], cluster\_id);