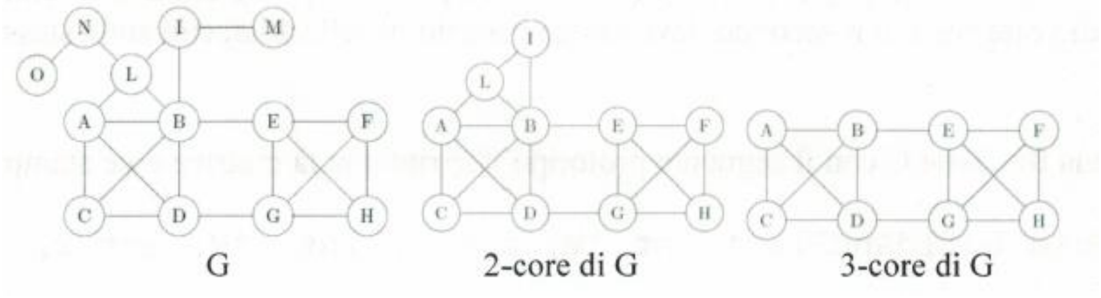
In graph theory, given an undirected, simple and connected graph G=(V,E):

- the maximal subgraph of G's vertices such that each one has degree ≥ k is called **k-core**. Only edges belonging to the subgraph are to be considered.

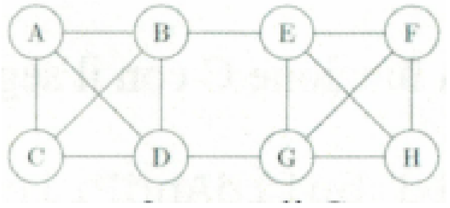
Example:



Algorithm:
to compute G's k-core all of his vertices with degree < k need to be deleted. Notice that a vertex removal imply the deletion of all the edges that are incident to it and this operation, in turn, decreases other vertices' degree, so they might now need to be removed. In the example, if k=2, the removal of the O vertex (degree=1) also makes the degree of N vertex equal to 1 (originally equal to 2), so it needs to be removed. The removal of the M vertex (degree=1) makes the degree of I vertex equal to 2 (originally equal to 3), so it doesn't need to be removed from the 2-core, but it will be removed from the 3-core.

- it is called **j-edge-connected** if and only if at least j edges need to be removed in order to disconnect it.

Example: the following graph is 2-edge-connected, by removing (B,E) and (D,G) edges it becomes disconnected. Instead, by removing a single edge the graph remains connected.



An undirected, simple and connected graph G=(V,E) is stored in a file by means of the list of its edges with the following format: on the first line there's an integer representing the |V| number of vertices of the graph, followed by |V| lines and each of them contains the alphanumeric ID of a vertex (max 10 chars) and at the end there is an undefined number of couples of ID representing the edges of the graph. The format of the file can be assumed to be correct.
Write a C program that receives the name of a file containing the G graph description as a command line argument and:

1. reads the G graph and stores it in an appropriate data structure;
2. reads keyboard input, an integer k ≥ 0, and computes, if it exists, the **k-core** of G, printing the list of vertices belonging to it;
3. reads keyboard input, an integer j ≥ 1, and verifies if G is **j-edge-connected** by excluding the existence of edges' set with cardinality < j able to disconnect it and finds at least an edges' subset of cardinality j able to disconnect the graph.

Comments and suggestions:
we suggest, whenever it is necessary to delete some vertices, to mark the removed ones instead of deleting them from the data structure of the graph. As a consequence, we suggest you to modify the graph's ADT of 1st class to consider this “logic” deletion.