Implement Kleinberg's HITS Algorithm, and Google's PageRank algorithm in Java, C, or C++ as explained.

(A) Implement the HITS algorithm as explained. Pay attention to the sequence of update operations and the scaling. For an example of the Subject notes, you
have output for various initialization vectors. For an explanation of the arguments see the discussion on PageRank to follow.
% java hits iterations initialvalue filename
% ./hits iterations initialvalue filename

(B) Implement Google's PageRank algorithm as explained. The input for this (and the previous) problem would be a file containing a graph represented through an adjacency list representation. The command-line interface is as follows. First we have the class/binary file (eg pgrk). Next we have an argument that denotes the number of iterations if it is a positive integer or an errorrate for a negative or zero integer value. The next argument initialvalue indicates the common initial values of the vector(s) used. The final argument is a string indicating the filename that stores the input graph.

% ./pgrk iterations initialvalue filename // in fact pgrkWXYZ
% java pgrk iterations initialvalue filename // in fact pgrkWXYZ

The two algorithms are iterative. In particular, at iteration t all pagerank values are computed using results from iteration t − 1. The initialvalue helps us to set-up the initial values of iteration 0 as needed. Moreover, in PageRank, parameter d would be set to 0.85. The PageRank of vertex A depends on the PageRanks of vertices T1, . . . , Tm incident to A, i.e. pointing to A. The contribution of Ti to the PageRank of A would be the PageRank of Ti i.e. PR(Ti) divided by C(Ti)), where C(Ti) is the out-degree of vertex Ti.

P R(A) = (1 − d)/n + d (P R(T1)/C(T1) + . . . + P R(Tm)/C(Tm))
The pageranks at iteration t use the pageranks of iteration t − 1 (synchronous update). Thus PR(A) on the left is for iteration t, but all PR (Ti) values are from the previous iteration t − 1. (In an asynchronous update, we have only one vector!) Be careful and synchronize!
In order to run the 'algorithm' we either run it for a fixed number of iterations and iterations determines that, or for a fixed errorrate (an alias for iterations); an iterations equal to 0 corresponds to a default errorrate of 10−5. A -1, -2, etc , -6 for iterations becomes an errorrate of 10−1, 10−2, . . . , 10−6 respectively. At iteration t when all authority/hub/PageRank values have been computed (and auth/hub values scaled) we compare for every vertex the current and the previous iteration values. If the difference is less than errorrate for EVERY VERTEX, then and only then can we stop at iteration t. Argument initialvalue sets the initial vector values. If it is 0 they are initialized to 0, if it is 1 they are initialized to 1. If it is -1 they are initialized to 1/N, where N is the number of web-pages (vertices of the graph). If it is -2 they are initialized to 1/√N. filename first.) Argument filename describes the input (directed) graph and it has the following form. The first line contains two numbers: the number of vertices followed by the number of edges which is also the number of remaining lines. All vertices are labeled 0, . . . , N − 1. Expect N to be less than 1,000,000. In each line an edge (i, j) is represented by i j. Thus our graph has (directed) edges (0, 2),(0, 3),(1, 0),(2, 1). Vector values are printed to 7 decimal digits. If the graph has N GREATER than 10, then the values for iterations, initialvalue are automatically set to 0 and -1 respectively. In such a case the hub/authority/pageranks at the stopping iteration
(i.e t) are ONLY shown, one per line.