Exercise 2

All the instructions related to running the code are given in README.txt file in the Assignment 3 folder.

Question 1

Code Explanation

I have implemented this with an example problem from pagerank class slides.

1. Imported the libraries numpy and fractions
2. Probability Transition Matrix M is defined. The values are taken from pagerank class slides
3. Now the number of nodes is M.shape[0] as dimension of M is nXn
4. An initial column vector v is defined such that it contains values 1/n
5. Now the next step is just the implementation of pagerank formula from class slides
   v_new = beta * M * v + (1 - beta)*e / n
   Where n is number of nodes i.e. M.shape[0]
   beta = 0.8
   e = $1/10^{10}$
6. Iterate from 1 to 100 and for iteration calculate the new v using the above pagerank formula and then see if the new_v converged.
   If the new_v does not converge then replace the current v with new_v and again repeat the process.
   Once the new_v converges, we break out of the loop.

7. After the loop ends v is the ratio of v with the sum of v
   I.e. v = v/np.sum(v)
8. Now the v contains the pagerank values and index is the node number
   So if v = [
   [pagerank_value1],
   [pagerank_value2],
   [pagerank_value3],
   [pagerank_value4]
   ]
   Then we can iterate through v using the enumerate method and print the node id and pagerank value.

```
(vpgrank) niketan@student-10-201-09-077 Exercise 2 % python pagerank.py
0 [[0.10135135137574423]]
1 [[0.1283783784139292]]
2 [[0.6418918917963973]]
3 [[0.1283783784139292]]
```

Question 2
Code Explanation
First we import the libraries required.
I have used scipy.sparse.coo_matrix instead of csc_matrix to convert the Probability Transition Matrix to a sparse matrix.

Import sys is used to take input from arguments provided in the command line
So filename = sys.argv[1]
The file contains the description of the file and links from FromNodeId to ToNodeId
Create empty dicts for incoming node id links and outgoing node id links
Also create an empty list of from node id and to node id.
Now read the file line by line and if line[0] that is the first character is # then ignore and continue.
Now split the from node id and to node id.
Example:
from_node_id_1 to_node_id_1
This means that there is a link from from_node_id to to_node_id
from_node_id → to_node_id
Once we get from_node_id and to_node_id, we increase the outgoing link for from_node_id by 1 and incoming link for to_node_id by 1.

Now append the list of from nodes and to nodes with the respective node ids.
This way we calculated the number of outgoing and incoming links for a particular node and stored the list of from node ids and to node ids.

The file does not contain all the nodes with incoming and outgoing links. To check this we check if the node has incoming and outgoing link. If it does not then print the nodeid.

This means that there are some nodes that are isolated.

Now the data that is considered for probability transition matrix is only the outgoing links.
So let's say if A has 3 outgoing links then the matrix value for A will be 1/3.
Now we create such list of data for all the number of outgoing links for a from_node_id.

Now we create a probability transition matrix M by using scipy.sparse.coo_matrix
With all the data with dimensions of all the to_nodes and from_nodes

As soon as we create the matrix M we then apply the same pagerank algorithm that we applied in Question 1 above.

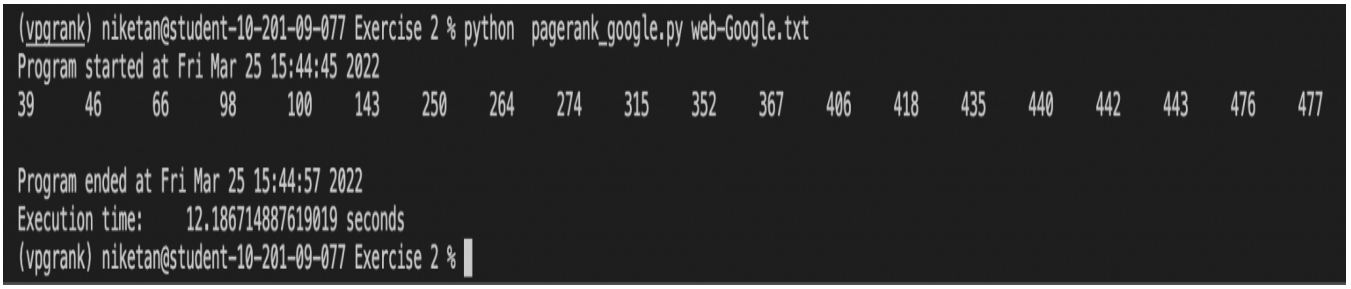
Once we get the matrix v with all the pagerank values we iterate through it and write the nodeid and pagerank values inside output_file.txt

We also maintain the pagerant_nodes_dict where nodeid is the key and pagerank of node is the value.

Then we calculate top 10 values using
top_10 = sorted(pagerank_nodes_dict, key=pagerank_nodes_dict.get, reverse=True)[:10]

This will search the dictionary and find out top 10 nodes with largest pagerank values.
Later we write all the top 10 nodes and next line with node id and its pagerank value inside the file top_10_output_file.txt

Execution time for Question 2



The execution time was approximately 12.19 seconds.