Big Data Applications- Individual Assignment.

This deliverable has 35% weightage in the Consolidated Total score.

INSTRUCTIONS:

1. This is an individual assignment. It has **three** questions.
2. All the relevant assignment files are in file: **Assignment.zip.**
3. You are free to create and run the programs on any cloud platform.
4. The assignment should be solved using **pyspark** without using any direct function calls for these algorithms. **Each problem should have a clearly defined map function and a reduce function.**
5. Any late submission will attract a **penalty** as mentioned in the course outline.
6. zip file submissions will not be accepted.
7. The honor code for this Assignment is **2N-C.**

DELIVERABLES: There will be 4 deliverables:

a. **3 Python Files with .py suffix** - **Please do not submit Jupyter notebooks**. Submit Python files which have a .py suffix which can be run using **python filename.py**; **DO NOT** use any **command line arguments**. Assume that all relevant input files/directories are within the same directory as the Python file.
b. **A report** - A report in **PDF** format called ***firstname_lastname_report.pdf*** describing your work and insight. **Complete Screenshot of the Output should be shown in the Appendix of the report.**

QUESTIONS:

1. **PageRank**

    a. You are given a graph in **input.txt**. Each line is an edge (source, destination) delimited using a tab.

    b. Use the basic PageRank formulation which we covered in class, i.e., the one without any jump probability.

c. The output format should be as follows: Every line should contain (VertexId, PageRank score) with a tab delimiter, and ascending sorted vertex ids.

d. Things to submit: *firstname_lastname _pagerank.py*

2. Inverted Index

a. You are given a file called **docs.zip**. Inside the docs directory there are seven text files: **d1.txt** to **d7.txt**. You should lower case each word and use the documents to generate an inverted index. **Do not remove any stop words.**

b. Create an inverted index which stores postings with frequency. Use filenames with .txt removed as document ids. You do not need to store position information in these postings.

c. The output format should be as follows: Every line should correspond to one word, such that lines are sorted in lexicographical order of words with the following format.

Word#docID:freq;docID:freq;docID:freq,…
e.g., weapons#d4:2;d5:1;d6:3;d7:3

d. Things to submit: *firstname_lastname _invertedIndex.py*

3. KMeans

a. You are given a file called **points.txt**. Each line contains information for a point: its two coordinates separated by a tab.

b. Run KMeans with K=3. The centroids should be initialized as (2,5), (6,2), (1,1).

c. The output format should be as follows: Every line should correspond to one cluster centroid. It should be in this format.

PointCoordinate1#PointCoordinate2
e.g., 4.738#1.745

d. **Things to submit: *firstname_lastname _kmeans.py***