1. Which Input Format did you use in the MapReduce program?

File Input Format that we use is a Text file and it's a combination of key-value pairs. As the input for reducer part we use a key and list of values.

2. What is the input and output format of the map function?

Input: Values of each line, Configuration context of the Job

Output: void. But writes data into a Tree Map

3. What is the logic of the map function?

It will get each line of the input, mix Id with two points of each all and consider it as a key. Calculate the distance of this points with the query points. Assign the computed distance as value to the key and write it to the Tree Map file.

4. If a combiner function is used, what is the signature of the combiner function (input and output) and what is its logic?

N/A

5. If a reduce function is used, what is the signature of the reduce function (input and output) and what is its logic?

It will get the values that we combined into or combiner and try to put them into a Tree Map. Every time we add a new entry into the tree it will check to see if it's exceeded the K factor. If it does, it will remove the highest distance (last key of the tree) which makes us having a sorted result as a tree.

6. How many mappers and reducers are needed for your program?

Since we are working with 445 MB of data and each bock of data is 128MB, we need about 3 or 4 mappers to get the best result. However since mappers are using a combiner, we are using only 1 reducer since we just want to sort the data and get the final result.

7. How many records are shuffled between the mappers and reducers?

Since in this implementation, I'm am not using any combiner, per each records of the input file, we will have same input for reducers. With that being said, we are going to have same number of shuffles as number of input records.

8. For the Pig Latin program, how many MapReduce jobs are needed to run the program? How does this compare to the MapReduce implementation?

Before the execution, Pig will decide about the execution plan and decide how many jobs are needed. In this case, since we doing multiple phases, as tested, Pig is deciding to have 4 MapReduce jobs for different phase of the program.

Compare to the MapReduce implementation, it seems that we have more control over MapReduce and we can come up with a better plan on how to run and compute this KNN problem. However, the amount of time that we spend and lines of codes that we wrote to implement Pig is significantly smaller than implementing a MapReduce program for the same purpose.