**1. What is the difference between TextInputFormat and KeyValueInputFormat class?**

**Ans :** The TextInputFormat class converts every row of the source file into key/value types where the BytesWritable key represents the offset of the record and the Text value represents the entire record itself.

The KeyValueTextInputFormat is an extended version of TextInputFormat , which is useful when we have to fetch every source record as Text/Text pair where the key/value were populated from the record by splitting the record with a fixed delimiter.

Consider the Below file contents,

AL#Alabama

AR#Arkansas

FL#Florida

If TextInputFormat is configured , you might see the key/value pairs as,

0 AL#Alabama

14 AR#Arkansas

23 FL#Florida

if KeyvalueTextInputFormat is configured with conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", "#") , you might see the results as,

AL Alabama

AR Arkansas

FL Florida

**2. How is the splitting of file invoked in Hadoop framework?**

**Ans :** How is the splitting of file invoked in Hadoop Framework ?

It is invoked by the Hadoop framework by running getInputSplit() method of the Input format class (like FileInputFormat) defined by the user

Consider case scenario: In M/R system,

- HDFS block size is 64 MB

- Input format is FileInputFormat

- We have 3 files of size 64K, 65Mb and 127Mb

then how many input splits will be made by Hadoop framework?

Hadoop will make 5 splits as follows

- 1 split for 64K files

- 2 splits for 65Mb files

- 2 splits for 127Mb file

**3. Consider case scenario: In M/R system, - HDFS block size is 64 MB - Input format is FileInputFormat – We have 3 files of size 64K, 65Mb and 127Mb How many input splits will be made by Hadoop framework for each file?**

**Ans :** 5 splits described as below.

- 1 split for 64K files

- 2 splits for 65Mb files

- 2 splits for 127Mb file

**4. After the Map phase finishes, the Hadoop framework performs “Partitioning, Shuffle and sort”. Explain each event in brief.**

**Ans :** - Partitioning

Partitioning is the process of determining which reducer instance will receive which intermediate keys and values. Each mapper must determine for all of its output (key, value) pairs which reducer will receive them. It is necessary that for any key, regardless of which mapper instance generated it, the destination partition is the same

- Shuffle

After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers. This process of moving map outputs to the reducers is known as shuffling.

- Sort

Each reduce task is responsible for reducing the values associated with several intermediate keys. The set of intermediate keys on a single node is automatically sorted by Hadoop before they are presented to the Reducer .

**5. What is a Combiner?**

**Ans :** The Combiner is a "mini-reduce" process which operates only on data generated by a mapper. The Combiner will receive as input all data emitted by the Mapper instances on a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers.

**6. What is Hadoop streaming?**

**Ans :** Streaming is a generic API that allows programs written in virtually any language to be used as Hadoop Mapper and Reducer implementations.

**7. What are the most commonly defined input formats in Hadoop and explain each in brief. 8. Explain what is distributed Cache in MapReduce Framework ?**

**Ans :** Following 3 are most common InputFormats defined in Hadoop   
- TextInputFormat  
- KeyValueInputFormat  
- SequenceFileInputFormat  
 **TextInputFormat** is the hadoop default.

**9. Explain what happens in textinputformat ?**

**Ans :** It reads lines of text files and provides the offset of the line as key to the Mapper and actual line as Value to the mapper.

**10. Explain what is Sequencefileinputformat?**

**Ans :** Hadoop’s sequence file format stores sequences of binary key-value pairs. Sequence files

are well suited as a format for MapReduce data because they are splittable (they have sync

points so that readers can synchronize with record boundaries from an arbitrary point in

the file, such as the start of a split), they support compression as a part of the format, and

they can store arbitrary types using a variety of serialization frameworks. (These topics are

covered in SequenceFile.)

To use data from sequence files as the input to MapReduce, you can use

SequenceFileInputFormat. The keys and values are determined by the sequence file, and

you need to make sure that your map input types correspond.