问题 1：在 Hadoop 中，一个处理文本文件的 MapReduce 作业，其 Map Task 数目是如何决定的？

（1）Hadoop 设置的 Split 与 Block 的关系来决定。

（2）输入文件的个数与大小

Map Task 的数目是由分片 Split 的数目决定的，Split 是 MapReduce 中最小的计算单元，一个 Split 对应一个 Map Task，而 Split 的个数又与 Block 个数有关。

默认情况下 HDFS 上一个 Block 对应一个 Split，例如（此处使用 Hadoop 新版本 Block size 默认情况为 128MB 进行说明）：

①输入的一个文件小于 128MB，默认情况下则保存在 HDFS 上的一个 Block 中，对应一个 Split 文件，所以将产生一个 Map Task。

②如果输入一个 129M 的文件就对应两个 Block，也就对应两个 Split，也就产生两个 Map Task。

③如果输入为三个小于 128M 的文件，就对应三个 Split，也就产生三个 Map Task。

与此同时，用户可自行指定 Block 与 Split 的关系，HDSF 中的一个 Block，一个 Split 也可以对应多个 Block。Split 与 Block 的关系都是一对多的关系。

问题 2：MapReduce 中 Partitioner 的作用是什么？

Map 阶段的输出，需要在 Shuffle 阶段进行分区和排序等操作。Partitioner 的作用实际上就是将 Map 每个节点的输出结果进行分类，通过 key 将每一个类的数据映射给不同的 Reduce Task 处理，所以 Partitioner 分的份数和 Reduce Task 的数量是相等的。默认情况下，MapReduce 使用的是 HashPartitioner，利用 key 的哈希值进行分区。用户可以继承 Partitioner，自定义分区的条件，实现自己的分区逻辑。

问题 3：输入集合如下表所示,按照以下表格的形式，填写 Map 阶段、Shuffle 阶段和 Reduce 阶段的输出,并简要描述各阶段的操作。

输入集合如下表所示,按照以下表格的形式

key_P	key_C
Tom	Jack
Jack	Alice
Jack	Jesse

Map 阶段：输入的 key 为行偏移量，value 为每行数据，value 值用空格分为两个字符串，前者为父母，后者为儿女。其格式为：<行偏移量,父母 儿女>

输出时 key 为父母则 value 为 child_儿女，key 为儿女则输出 parent_父母。其格式为：<父母,child_儿女>或<儿女,parent_父母>。

Map 阶段	
输入	输出
<0,Tom Jack> <8,Jack Alice> <18,Jack Jesse>	<Tom,child_Jack> <Jack,parent_Tom> <Jack,child_Alice> <Alice,parent_Jack> <Jack,child_Jesse> <Jesse,parent_Jack>

Shuffle 阶段：将 key 相同的键值对进行合并。

Shuffle 阶段	
输入	输出
<Tom,child_Jack> <Jack,parent_Tom> <Jack,child_Alice> <Alice,parent_Jack> <Jack,child_Jesse> <Jesse,parent_Jack>	<Tom,[child_Jack]> <Jack,[parent_Tom,child_Alice,child_Jesse]> <Alice,[parent_Jack]> <Jesse,[parent_Jack]>

Reduce 阶段：假设一共有 n 代人(n≥3)，则核心是找到家族中的第 k 代人（2≤k≤n-1），因为 k 代人才有可能既有父母又有儿女（不取叶节点和根节点），即在 Shuffle 阶段输出的结果中 value 中既有 child 又有 parent（既有前驱又有后继）。扫描所有的 key，准备两个临时数组 pro 和 next，扫描 value 中的内容，parent_开头的写入 pro，child_开头的写入 next，pro 和 next 中的内容两两组合写入 context，清空 pro 和 next 数组，重复操作，直到遍历完所有 key。

在本题中只有三代人，只有 Jack 是第二代人，Tom 写入 pro 数组，Alice、Jesse 写入 next 数组，两两组合后的结果为<Tom,Alice> <Tom,Jesse>。

Reduce 阶段	
输入	输出
<Tom,[child_Jack]> <Jack,[parent_Tom,child_Alice,child_Jesse]> <Alice,[parent_Jack]> <Jesse,[parent_Jack]>	<Tom,Alice> <Tom,Jesse>