/\*

\* Type Pre-Requisites.

Author: Anbalagan Mookkaiah

Requirements:

1. Ignore Caps while counting. i.e Who and who are same for counting purposes

2. For reporting, purposes change every word to lower case

3. For reporting, remove hyphen. Hi-lite should be hilite

4. For reporting, provide a tab between key and value

5. For reporting, sort by keys (in alphabetical order)

Assumption - all characters are ASCII characters

/\*

\* Using Hadoop, I could configure Distributed File System block size and the number of map tasks / node based on available node in the cluster.

\* Generally the right level parallelism seems to be around 10-100 maps per-node.

\* For example, 10TB of input data with blocksize of 128MB, ends up with 82,000 maps.

\* Number of reduce tasks can be configured such as [0.95 \* Number of node].

\* Keeping slightly less than the whole number makes few reduce tasks available for failure tasks.

\* Linear Scalability:

\* When more resources and loads are added, could should be able to handle with lineatly increased output.

\* Mutable state protected with lock. Reduce lock duration.

\* lock stripping / putIfAbsent / replace / ConcurrentHashMap.

\* read write lock.

\*

\* Atomic Variable / Imutable objects (final, private)..

\* Better GC.

\* ConcurrentHashMap / TreeMap

\* Assumed 64 bit Solarice that allocated 1 MB stack / Thread.

\*

\* The limitations are generally based on system resources -

physical memory and CPU.

The number of threads that can run within a JVM process is generally

limited by the address space for that process. Each thread requires

a thread stack. The more threads you have, the more process address

space you use. The more address space you use for thread stacks, the

less you have for the Java heap. There is one more tricky you have when you have less heap;

it triggers more GC occurance. This effects performance largely. So there's a bit of a balance

that needs to be achieved. Generally, if you have a lot of active threads you will probably want

a lot of heap space as well.

Handle Memoty leak..

number of threads = total virtual memory / (stack size\*1024\*1024)

-- cat /proc/sys/kernel/threads-max - Shell script to find number of threads allowed.

public class ThreadReader {

public static void main(String[] args) {

File f = null;//folder

final BlockingQueue<File> queue = new ArrayBlockingQueue<File>(1000);

for(File kid : f.listFiles()){

queue.add(kid);

}

ExecutorService pool = Executors.newFixedThreadPool(5);

for(int i = 1; i <= 5; i++){

Runnable r = new Runnable(){

public void run() {

File workFile = null;

while((workFile = queue.poll()) != null){

//work on the file.

}

}

};

pool.execute(r);

}

} \*/