**Intro to Cloud Computing**

## Homework 4

***MapReduce in mincemeatpy***

*Due 11:59pm March 30 (Wednesday night after break)*

*Each student should submit the github.uc.edu SSH clone address of their repository to Blackboard, along with the commit hash to grade.*

This homework assignment will let you explore how to convert simple iterative algorithms to mapreduce in the mincemeatpy python framework. Use either the official version, or my own <https://github.com/fuzzpault/mincemeatpy>

Each of the 3 programs should be labeled with the exact filename given below. All programs MUST use mincemeatpy for all data analysis! Use only 1 mapreduce pass. Assume mincemeatpy is in the current directory and clients will be running when your program is run. Set the password to ‘changeme’.

All 3 files should be submitted via github.uc.edu as a private repository. Please make me [paul.talaga@uc.edu](mailto:paul.talaga@uc.edu), [backjo@ucmail.uc.edu](mailto:backjo@ucmail.uc.edu), and [kaneca@mail.uc.edu](mailto:kaneca@mail.uc.edu) are contributors so we can clone. Include the git clone URL in your Blackboard submission, as well as the commit SHA value we should use.

There are 3 programs/scripts to write for this assignment:

1. Math stats - mathstats.py - Given a text file name on the command-line containing one number per line, print out the sum, count, and standard-deviation of all the numbers in the file. All statistics should be found in one pass through the data.
2. Palendrome prime number finder - primes.py - Output all the prime numbers which are palindromes between 2 and 10 million. A palindrome number is one which when read backwards is the same value. Ex. 919 is prime and a palindrome.
3. Password cracking - passCrack.py - Given a string of characters on the command line, find what string hashes to it. Passwords are sometimes stored in a hashed form, so if the database is breached, the passwords are not easily usable. For this assignment, assume we have a hash of a password in hex form. Given this hash on the command line, find what password hashes to it. Only the first 5 characters of the hash are checked. Assume passwords are 4 or fewer characters containing only lowercase letters and numbers. Use MapReduce to quickly look through all combinations for a match. Print out the input hash string and the valid passwords which hash to it, if any. Use hashlib md5 hexdigest()and use the first 5 characters. Here are some passwords to crack: d077f, 0832c, 1a1dc, ee269, 0fe63

### 

### Your grade will consist of the following:

1. 30% - mathstats.py 9% per sum, count, standard-dev, 3% one pass.
2. 30% - primes.py - Finds all values.
3. 40% - passCrack.py - finds the password.

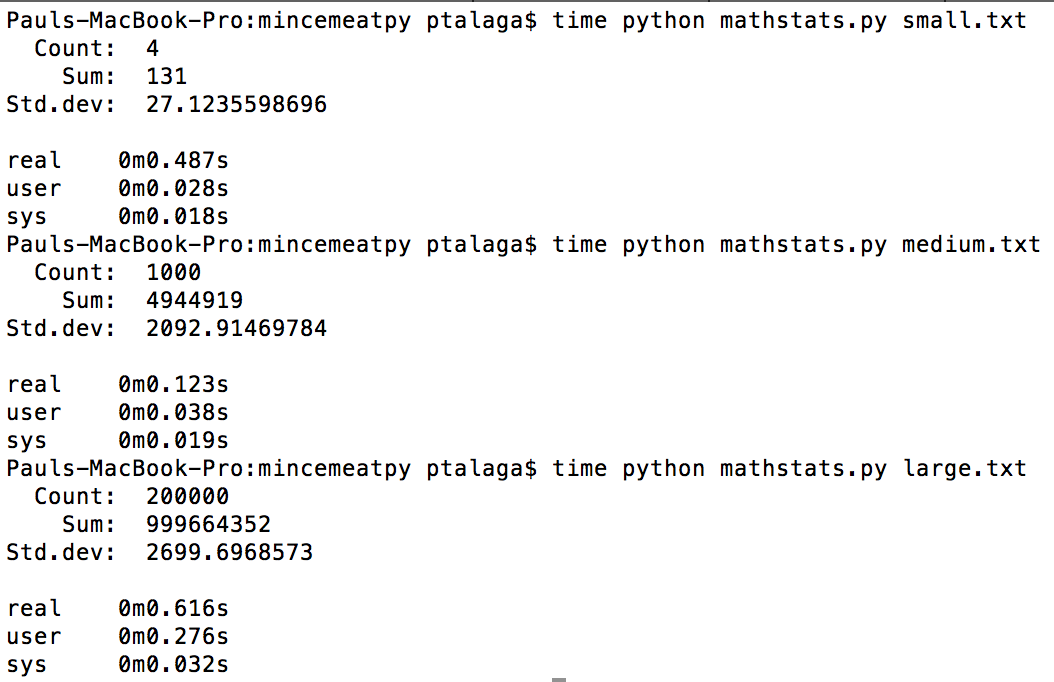
2% extra credit if primes.py can find all values in under 2 minutes. 2% extra credit if passCrack.py can find a solution in under 2 minutes. Assume at least 4 worker clients.

### Notes:

1. ALL result data must go through map-reduce. If any data does not, no credit will be given. For example if you read in numbers from a file, save the count of numbers, and then use that later to calculate the mean, that is a violation. Some calculation based on the resulting reduce key/values is allowed such as counting the number key/value pairs, but put as much work into map and reduce as possible.
2. Think about the granularity of parallelism. Tuning the size of the jobs is key to fast execution.
3. All evaluation will be done with at least 4 mincemeatpy worker nodes.

**Example Output:**

mathstats.py



passCrack.py

time python passCrack.py d077f

Attacking d077f

{'found': ['cat', 'gkf9']}

real 0m3.116s

user 0m1.803s

sys 0m0.123s

time python passCrack.py 0832c

Attacking 0832c

{'found': ['cats', 'zb8e']}

real 0m2.846s

user 0m1.831s

sys 0m0.124s