Problem 1

(a)

```
-----------------------------------------
Cracking easy hash...
Hash: ef0ebbb77298e1fbd81f756a4efc35b977c93dae
Password: orange
Took 124 attempts to crack input hash. Time Taken: 0:00:00.081462
-----------------------------------------
```

(b)

```
-----------------------------------------
Cracking medium hash...
Hash: 0bc2f4f2e1f8944866c2e952a5b59acabd1cebf2
Password: starfish
Took 2681 attempts to crack input hash. Time Taken: 0:00:00.071932
-----------------------------------------
```

(c)

```
-----------------------------------------
Cracking leet-hacker hash...
Hash: 9d6b628c1f81b4795c0266c0f12123c1e09a7ad3
Password: puppy
Took 5638 attempts to crack input hash. Time Taken: 0:00:00.004354
-----------------------------------------
```

(d)

```
-----------------------------------------
Cracking extra-credit hash...
Hash: 44ac8049dd677cb5bc0ee2aac622a0f42838b34d
Password: z745100 wujuchawiapra53
Took 230983159156 attempts to crack input hash. Time Taken: 14:03:36.706914
-----------------------------------------
```

Method:

(a)(b)(c) just use hashlib to bruteforce, (c) did it twice to find the plaintext of salt first then the hash of the password.

(d) I used **multiprocessing** to fork processes according to the number of CPU core and put it on the CC's computer. Each process handle its own portion of combinations (10^12/#CPU) to make sure there's no repetition.

Problem 2

name	speed(s)	hash value
sha1	0.13800048828125	b29ae9b33d33304b3b966f2921cc5bfb3cb3c3ce
md5	0.18719077110290527	cab08b36195edb1a1231d2d09fa450e0
sha512	0.19601035118103027	e6eaef73af4b739daf7e8874e1f3b87b4d320f954347e912c6cbb33f686c428b94832c46f7928e9cf685e14452f5a0e3209edae501ac222fa6eaae7dbbb7488a
sha256	0.2950263023376465	1cadc5e09cbb81044e256f9fc67090fcf86d7a596145eb615844fe15341451e6
sha224	0.3002297878265381	2dd11ca85546f0bf1029299f5d38383ab0f0942b61ae1b92b5a384be
sha3-224	0.3444499969482422	26c55e271dc576d3db2653dc952ab5303cc521ff788acd63a9f16716
sha3-256	0.3563575744628906	02db744889e01a17accabbb69a0eca49a39058ed560d673170c631f096bef1be
sha3-512	0.6621861457824707	58d0bc115ddaa7a8a03245b054be6e9b59d338508d00313b486b81430f51514c1ca5b3d569093ea795e0d97c2c17861925af55250fff5a4a2250b5897d381dba

(a) In attachment.

(b) We can observe that sha1 is the fastest.

(c) Sha1>md5>sha512>sha256>sha224>sha3-224>sha3-256>sha3-512

Problem 3

```
row: 1, col: 98
diff_sum: 0.20000000000000284
row: 2, col: 49
diff_sum: 3.0
row: 7, col: 14
diff_sum: 4.6000000000000005
row: 14, col: 7
diff_sum: 7.799999999999999
row: 49, col: 2
diff_sum: 26.999999999999993
row: 98, col: 1
0 1 2 3 4 5 6
T H E Q U E S
T I O N O F W
A G E A N D P
R I C E C O N
T R O L S W I
L L H A V E T
O B E F A C E
D I N S I X T
Y E I G H T I
F C O N G R E
S S D O E S N
O T A P P R O
V E A T A X I
N C R E A S E
```

Process: I've tried using the technique mentioned in the hint (find the less sum of

difference of expected number of vowels). However, I've spent so much time on trying to crack 7*14 (as it seems to be the most reasonable one). I even used the English dictionary to find some clues and eventually give up. Then I gave 14*7 a try and... it is actually quite simple to discover the plaintext. I've learned that maybe sometimes we cannot trust all the things in the hint (?

Plaintext:

THE QUESTION OF WAGE AND PRICE CONTROLS WILL HAVE TO BE FACED IN SIXTY-EIGHT IF CONGRESS DOES NOT APPROVE A TAX INCREASE

You can simply use python <problem.py> to run all the problems. Note that the output of problem1 will be recorded in the result.txt file. Problem 1-d have to take hours or days (depends on your computer) to crack.