Assignment: Password cracking - the brute force way

In this assignment, you are only to apply a centralized application to crack passwords and to discuss a distributed application to crack passwords! Using more cores or computers should make the program run faster.
The goal is to get a large speed-up: That is to see how much faster you can make the program run, compared to the centralized (non-distributed) version of the program.

Centralized (non-distributed) version of the program

The general idea is that you have a password file with usernames and password_hash_values (nicknamed encrypted passwords) separated by ":" e.g. (username:password_hash_value).

You also have a large dictionary (list of words) that users *might* have used as passwords. The words of the dictionary are run through a Hash Function to generate a "fingerprint"/hash_value, then this "fingerprint" is compared with each password_hash_value from the password file. If you have a match, you have found a password, now in clear text.

Some users might not use an exact word from the dictionary, but may have made some kind of change to the words (transformations), like

- Starting with a capital letter
- All capital letters
- Any arbitrary number of capital letter of the beginning of the word
- Adding 1 or 2 digits to the beginning of the word
- Adding 1 or 2 digits to the end of the word
- any combination of the above.

Here are given some examples of the changes from above (Be aware not all these transformations are implemented in the centralized version):

- secret
- Secret
- SECRET
- SECret
- 5secret
- secret3
- Secret123

As can be seen from the text-books many users chose passwords which are more or less based on a word from a dictionary.

Dictionaries

The centralized version project includes a dictionary "webster-dictionary". For example inside the bin/debug in the Visual studio project.

Hint: During developement you might use a reduced dictionary "webster-dictionary-reduced.txt" to cut down the executing time.

Password file

The centralized version project also includes a password file. Each line in the file contains username + password_hash_value.
The passwords hash_values are encoded using BASE64 encoding to make them into text strings storable in a text file.
You can actually also try a few other dictionaries.

Getting started

To get you started you must download a centralized C# version of the password cracker or the java version centralized Java version of the password cracker from your teachers home page.

Assignment 0
For all computers in the group run Centralized version. Record the time together with the computer-speed, number of cpu's and other relevant parameters. Now you know who should be the server!

Assignment 1
Discuss the Centralized version making sure everybody understands the program.

Assignment 2
Discuss an architecture of the distributed architecture type 1, 2a or 2b.
See below.

Different architectures

The general idea can be implemented using distributed architectures.

- master/slave architecture (for detailed information see Master-Slave)

1. Master / Slave where the master and the slaves are running in separate threads on one computer. . They can use shared memory.
2. Master / Slave where the master and the slaves are running in separated processes, e.g. on different computers communicating over TCP sockets.
   This comes in two variations:

a. The master is the server. The slaves are clients. Each client/slave ask the master/server for a part of the job.
b. The master is a client. The slaves are servers. The master ask each slave/server to do a part of the job.

The variation 2a. is the realistic one and variation whereas the architecture in variation 2b. is more simpel.

Each architecture can be implemented using threads (useful if the computer has more than one CPU) and by using processes with socket for communication. You only choose one architecture to work with.