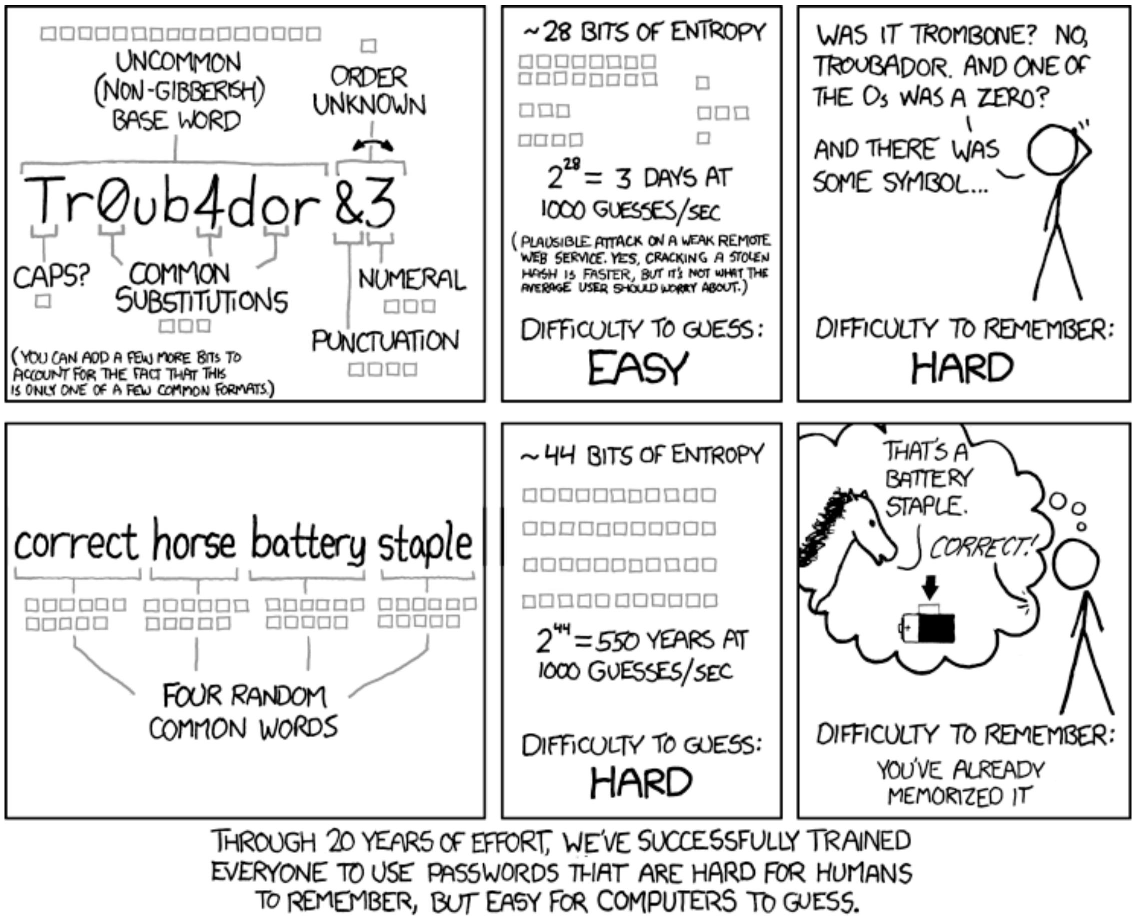
CSCI 136 Programming Exam #0
Fundamentals of Computer Science II
Spring 2014

This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #0 dropbox. Please ***double check you have submitted all the required files***.

You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

http://xkcd.com/936/

Overview. You are developing a new operating system. You need to develop classes to track user accounts in the system, verify login credentials, and track users' disk quota and successful login attempts. You are also concerned about the security of your system, so you will also develop an account cracking tool. This tool will measure how many accounts can be cracked in your system by guessing credentials from a list of common usernames and a list of common passwords.

To get started, download the zip file containing stub classes as well as a variety of test data files:

http://katie.mtech.edu/classes/csci136/users.zip

Part 1. Develop a class `User` that represents an individual user account in the system. A `User` knows things like the username of the account, the password of the account, the current disk usage of the account, and the total number of successful logins into the account. Here is the public API (the stub code has detailed descriptions of what each method does):

```
public class User
-----------------------------------------------------------------------------------------
        User(String username, String password, double diskSpaceUsed, int previousLogins)
 String getUsername()
    int getLogins()
boolean login(String password)
 double getDiskQuotaPercent(double diskQuota)
```

Part 2. Develop a class `UserDatabase` that stores a collection of user accounts. A `UserDatabase` knows things like all the user accounts in the database and the disk usage allowed per user (all users have the same quota). Here is the public API (the stub code has detailed descriptions of what each method does):

```
public class UserDatabase
-----------------------------------------------------------------------------------------
            UserDatabase(String filename, double diskQuota)
        int getNumberUsers()
LoginResult login(String username, String password, boolean printMessages)
```

A user database is stored as a simple text file where each line represents one user in the system. Each user has a username, password, current disk usage (a floating-point), and a count of previous successful login attempts (a non-negative integer). Here is `users5.txt`:

```
jacob      123456     0.0     0
sophia     password   0.0     0
william    iloveyou   23.34   23
ava        Adobe123   174.32  27
matthew    AZERTY     9.89    98
```

Note: *Passwords are case sensitive!* So the user "ava" must authenticate with the first letter of her password capitalized. On the other hand, *usernames are NOT case sensitive*.

The `login` method in `UserDatabase` can optionally be set to print informative messages to standard output about a given login attempt. Here are the three possible types of message along with an example:

- *Username not found.* Example message: "Username 'jake' not found!"
- *Username found, password incorrect.* Example message: "Password was not correct!"

- *Username found, password correct.* Displays the current percentage of disk space used (to two decimal places) as well as the number of successful login attempts (including this one). *Helpful tip:* You can output a literal percent symbol in `System.out.printf()` by using two percent symbols in a row, "%%". Example message: "Login successful, disk quota used 9.89%, total logins 99"

We have provided an interactive test `main` method in `UserDatabase`. You should NOT need to modify this program. Here are some example runs:

```
% java UserDatabase
UserDatabase <filename> <disk quota>

% java UserDatabase users5.txt 100.0
Loaded database of 5 users
Username (type "quit" to exit) : jake
Password : password
Username 'jake' not found!
Username (type "quit" to exit) : jacob
Password : 123456
Login successful, disk quota used 0.00%, total logins 1
Username (type "quit" to exit) : jacob
Password : 123456
Login successful, disk quota used 0.00%, total logins 2
Username (type "quit" to exit) : ava
Password : adobe123
Password was not correct!
Username (type "quit" to exit) : ava
Password : Adobe123
Login successful, disk quota used 174.32%, total logins 28
Username (type "quit" to exit) : matthew
Password : azerty
Password was not correct!
Username (type "quit" to exit) : matthew
Password : AZERTY
Login successful, disk quota used 9.89%, total logins 99
Username (type "quit" to exit) : quit

% java UserDatabase bogusfilename 100.0
Failed to open file: bogusfilename
Loaded database of 0 users
Username (type "quit" to exit) : quit
```

Part 3. Now for the fun part! You are going to develop a command-line tool `PasswordCracker` that makes use of your `UserDatabase` object to measure how many accounts in your system can be cracked by brute force. The tool requires three command-line arguments. If it doesn't receive this many, it prints out a helpful error message and terminates:

```
% java PasswordCracker
PasswordCracker <user database> <common usernames file> <common passwords file>
```

The user database file is the same format as in part 2. We don't care about disk quota in the password cracking tool (you can set the quota to any value you like). If the user database is not found, your program terminates without proceeding any further:

```
% java PasswordCracker bogus usernames-92k.txt passwords-10k.txt
Failed to open file: bogus
```

You have downloaded from the Internet a list of 92K common first and last names. Each name appears on a separate line, here is the start of `usernames-92k.txt`:

```
aaron
abbey
abbie
abby
abdul
...
```

You also downloaded a list of 10K common passwords. Each password appears on a separate line, here is the start of passwords-10k.txt:

```
password
123456
12345678
1234
qwerty
...
```

If either the username or password file is not found, print an error message and terminate:

```
% java PasswordCracker users5.txt bogus passwords-10k.txt
Failed to open username file: bogus

% java PasswordCracker users5.txt usernames-92k.txt bogus
Failed to open password file: bogus
```

You brute force cracking algorithm works by trying all possible usernames. If a username is in the system, it then tries all possible passwords (until it either finds the right one or tries all passwords in the common password file). During the search, you keep track of the total number of account usernames you guessed correctly, and how many of these were successfully broken into by guessing a correct password.

Unfortunately, the passwords file you downloaded only contains passwords in lowercase. You suspect you could crack more accounts by also guessing various case variants. Thus attack each account with:

- The original lowercase password from the common passwords file.
- A completely uppercase version of the password (e.g. "qwerty" -> "QWERTY").
- A version of the password where the first letter is capitalized (e.g. "qwerty" -> "Qwerty").

You program should display the username and password of any successful attack. It should NOT display other messages about the login process or the user's disk quota. At the end, your program should print how many usernames were found (both the absolute number and as a percent of the total system users). Similarly, it should print the number of accounts that were cracked (both the absolute number and the percentage). Percentages should be output to two decimal places. Here are some example runs:

```
% java PasswordCracker users5.txt usernames-92k.txt passwords-10k.txt
Cracked: sophia -> password
Cracked: william -> iloveyou
Cracked: jacob -> 123456
Cracked: matthew -> AZERTY
Usernames found  : 5 (100.00%)
Accounts cracked : 4 (80.00%)
```

NOTE: These lines should all appear, but they don't necessarily need to appear in this order. You should however get the same final stats.

```
% java PasswordCracker users25.txt usernames-92k.txt passwords-10k.txt
Cracked: elijah -> 000000
Cracked: ethan -> abc123
Cracked: daniel -> password1
Cracked: sophia -> password
Cracked: william -> iloveyou
Cracked: elizabeth -> 12345
Cracked: olivia -> 1234567
Cracked: alexander -> monkey
Cracked: noah -> 111111
Cracked: jacob -> 123456
Cracked: ella -> trustno1
Cracked: emma -> qwerty
Cracked: mason -> 12345678
Cracked: madison -> shadow
Cracked: chloe -> princess
Cracked: matthew -> AZERTY
Cracked: emily -> Admin
Cracked: mia -> 1234
Cracked: isabella -> 123456789
Usernames found  : 22 (88.00%)
Accounts cracked : 19 (76.00%)
```

```
% java PasswordCracker users1000.txt usernames-92k.txt passwords-10k.txt
...
Cracked: massimo -> cbr900rr
Cracked: elmer -> MUFASA
Usernames found  : 387 (38.70%)
Accounts cracked : 387 (38.70%)
```