**CSCD240 Homework 5 – Low-Level File I/O and System Calls**

**1, Problem Description**

Based upon the discussion in the class regarding how Unix system stores users’ name and password and performs the password-based access control, in this project, each student pretends to be a “hacker”. After the “hacker” (i.e. the student) obtains the file that contains users’ password in an encrypted format, named as **pass.txt**, the “hacker” (i.e. the student) is required to decrypt the password and write the decoded original password into another text file, named as **plainpass.txt. Note that in this project, we assume user’s password consists of only 4 lower-case letters, which will reduce your searching space and expedite your program execution.**

**2, What is provided?**

1. All demo code about low-level file I/O is provided on EWU Canvas. The demo code for password generation and authentication using System Calls are provided as well.
2. You can also find online references at URL:

<http://www.gnu.org/software/libc/manual/html_node/crypt.html>

1. The input file named as **pass.txt** is provided, whose format is shown as follows. **Each line** in this text file is an encrypted password. Here, we have a total of three passwords stored in this file. (The file pass.txt has been included, so you do not need to copy from here.)

$1$6gMKIopE$I.zkP2EvrXHDmApzYoV.B.

$1$pkMKIcvE$WQfqzTNmcQr7fqsNq7K2p0

$1$0lMKIuvE$7mOnlu6RZ/cUFRBidK7PK.

**3, What to do?**

1. You are required to read in the pass.txt into memory **using low-level I/O system call** functions.
2. You have to exhaustively generate all possible groups of 4 lower-case letters. Once you generate one group of 4 letters, such as “bike”, you pass it into crypt() function. If the string returned by crypt() function matches one line ***L*** in the pass.txt file, that means the line ***L*** is the encrypted message for plain password “bike”, thus the message *L* is successfully decrypted.
3. You have to decrypt all three lines of messages in the pass.txt file.
4. You have to output all decrypted password into another file named as **plainpass.txt by using low-level file I/O system calls.** The file format is shown as below. Each line in the file plainpass.txt is a decoded original password for the **corresponding** encrypted message in pass.txt. As we know before, the password consists of 4 lower-case letters, each of which is shown by a symbol ‘x’ below**.**

xxxx

xxxx

xxxx

1. Your program has to solve the whole problem in less than 5 minutes on cslinux. Hint: after all three messages have been decoded, your program stops searching.
2. You are required to include a simple makefile to compile your source code into an executable **hw5**.
3. You are required to deallocate all memories you dynamically allocated in your program. In addition, you are required to check your memory deallocation **by running your program with the valgrind command on the cslinux machine**. **You are required to create a pdf file that shows the valgrind memory-check result. If valgrind shows a memory leak, you lose 20% of total points.**