CSC 101 Lab Weeks 8 and 9
Security Lab

ISSUED: Friday, 18 May 2012
DUE: Lab 8: Wednesday 23 May 2012, by the end of lab
Feedback: Friday 25 May, during lab
POINTS POSSIBLE: 1
WEIGHT: 1% of total class grade

Specification of Program Caesar

This assignment requires you to write a program that will encipher text using the basic Caesar cipher. This cipher is fairly simple to implement, but the point of the assignment is to write code that is as robust as possible.

The Caesar cipher is an alphabetic shift cipher, which is a type of substitution cipher. This is the simplest and most widely known cipher. It works simply by shifting the alphabet by a specific shift size so that with a shift of 4, an 'a' in the plaintext would be replaced by a 'd' in the ciphertext.

Example:

```
Shift size: 6
Plaintext (P):    TOBEORNOTTOBE
Cipher(C):    ZUHKUXTUZZUHK
```

Details of the Program

- Caesar takes a number representing the shift size to encipher with, and input and output filenames. If a dash is present for either filename, stdin or stdout is used (as appropriate). If there are no command-line arguments, the program will read them from standard in.

```
Usage:  caesar shift infile outfile
```

- You must store these three command-line arguments in a struct. The definition details of the struct are up to you.
- Caesar will encipher its entire input until it reaches the maximum number of characters, which should be defined as 100.
- Caesar must check its command-line options for validity. If they are invalid â e.g. a shift that is not a number â it should print a usage message and terminate gracefully with nonzero status.
- If caesar succeeds, its exit status should be zero. If unsuccessful, it should be nonzero.
- The cipher only operates on normal alphabetic characters ("A-Za-z"), and all lower-case letters are mapped to upper-case.
- Anything that is not "A-Za-z" in the input is passed through unchanged. (Were this a real enciphering tool, these characters would either be enciphered, too, or dropped.)

Sample Runs

If there are no command-line arguments, then the program will prompt for and read three values from stdin, like you did in lab 7. For example, here's what it looks like with three command-line args:

```
caesar 6 intext ciphertext
```

where "`intext`" and "`ciphertext`" are the names of files. If you run the program without command-line arguments, then it will look like this:

```
caesar
Shift Size: 6
Plain Text File: intext
Cipher Text File: ciphertext
```

Again, this is just like what you did in lab 7, where the program either reads command-line args, or inputs from stdin if there are no command-line args.

Resources

1. Useful library functions, constants, and variables:
   - `int isalpha(int c)` -- defined in ctype.h
   - `int toupper(int c)` -- defined in ctype.h
   - `exit` -- defined in stdlib.h
   - `FILE* fopen(char* filename, char* mode)` -- defined in stdio.h; used in lab 5
   - `FILE* fdopen(int filedes, char* mode)` -- defined in stdio.h; like fopen but uses file descriptor instead of string filename
   - `STDIN_FILENO`, `STDOUT_FILENO` - file descriptor constants defined in unistd.h; use as first argument to `fdopen`
   - `void perror(char* s)` -- defined in stdio.h
   - `errno` -- library variable defined in stdio.h, and used in conjunction with the `perror` function
2. You can read the UNIX man pages for any of the preceding functions. For example, to read the man page for `perror`, type the following on a terminal:

   ```
   man perror
   ```

   You can read man pages in emacs by typing `<escape>x man`, and then entering the name of the man page after the "`Manual entry:`" prompt.
3. Some resources on robust programming will be discussed during the Friday lab presentation, and you can do your own internet search on the subject. Here are some specific suggestions from Tanya:
   - Robust programming: http://nob.cs.ucdavis.edu/clinic/
   - Bruce Schneier Blog: http://www.schneier.com
   - Read the news!

Collaboration

Unlike previous labs, you will work individually on labs 8 and 9.

Submitting and Receiving Feedback on Your Work

On or before the end of lab on Wednesday May 23, submit your work as follows:

```
handin gfisher 101_lab8 caesar.c
```

During the lab on Friday May 25, you will meet with security clinician Tanya to examine potential flaws in robustness and what to do to fix the flaws. Remember, you are trying to catch these things before she does, but the important lesson to learn is how to fix them. You will be graded on participation in the clinic and completeness of the second version of the program, which will be the deliverable for Lab 9.

Lab Schedule for Weeks 8 through 10

- ***Friday 18 May:*** lecture on security, from Tanya Luthin
- ***Monday and Wednesday, 21 and 23 May:*** work on caesar, and ask Tanya questions
- ***Friday 25 May:*** Feedback from Tanya, and assignment of work for Lab 9
- ***Monday 28 May:*** holiday
- ***Wednesday 30 May:*** lab quiz

• ***Friday 1 June:*** finish lab 9