

**CmpE102**  
**Spring 2017**  
**Programming Assignment 3**

**Due Date:** 13 Mar. 2017

**Goal:** ROT13 program

**Instructions:**

**Step 1. Creating the assembly language file**

Everything should be done the same way as Assignment 1.

**Step 2. The application area**

Our next programming assignment has to do with ROT13 encoding. ROT13 is an example of a simple substitution cipher called a *Caesar cipher* (because it was supposedly used by Julius Caesar 2000 years ago). This site will tell you more about [Caesar ciphers](#).

ROT13 is simply a Caesar cipher with a rotation of 13. Here's a typical definition:

rot13 /rot ther'teen/ /n.,v./ [Usenet: from `rotate alphabet 13 places'] The simple Caesar-cypher encryption that replaces each English letter with the one 13 places forward or back along the alphabet, so that "The butler did it!" becomes "Gur ohgyre qvq vg!" Most Usenet news reading and posting programs include a rot13 feature. It is used to enclose the text in a sealed wrapper that the reader must choose to open -- e.g., for posting things that might offend some readers, or spoilers. A major advantage of rot13 over rot(N) for other N is that it is self-inverse, so the same code can be used for encoding and decoding.

In other words, the advantage of using ROT13 rather than ROTn where  $n \neq 13$  is that since there are 26 letters in the alphabet,  $n = 13$  is the only value that can use the same algorithm for encoding and decoding.

- Here is a page where you can [try out ROT13](#).

Now you understand what ROT13 is, and you have a way to get correct examples of ROT13 encoding. What should your code do?

### Step 3. What your code should do

1. Your code will look very similar to Dunteman's **uppercaser** code that we went through in class.
2. Your code should use STDIN and STDOUT for input and output. (This is the default.) Use redirection on the command line to read from a file and write to a file.
3. Your code should open a file, read it character by character and output each character in ROT13 encoding.
4. Assume that the input file contains just ASCII text. Don't worry about what happens with non-text files.

### Hints

1. There are really two different parts to this problem. One is how to do I/O, the second is how to do ROT13 encoding. Solve one problem at a time.
2. Start with how to do I/O. That will be very similar to the **uppercaser** code.
3. Once you've got that figured out, you can do the ROT13 encoding part.
4. Note that ROT13 only changes the values 'a' through 'z' and 'A' through 'Z'. All other characters remain unchanged.

### Step 4. Turning in the assignment

We will try skipping the time-consuming in-class demonstrations. Turn in the following:

1. Commented listing file.
2. Demonstration of encoding an input text file of at least 100 characters. (Grab a piece of text off a website.) Show input and output characters.
3. Run the program a second time to show that the encoded output from the previous run can be decoded back to the original text.

Your assignment cannot be graded if you don't submit evidence that it functions correctly, i.e., a successful encoding and subsequent decoding of a text sample.