Assignment 9, Fall 2015
CS 4630, Defense Against the Dark Arts
Wahoo Virus

Purpose
This assignment illustrates the power of very-high-level languages and program generators. We will add a new pattern to our pattern file from the previous assignment to detect the “Wahoo” Virus.

Due Date
This assignment is due on Wednesday, 12/02/2015, at 10:00am

New Virus
Some bad actors that took Defense Against the Dark Arts a while back have used their knowledge to unleash a new virus called the Wahoo virus. The Ministry of Magic (MOM) has asked that we respond to this looming threat quickly.

The Wahoo virus uses encryption to make analysis more difficult. It also uses anti-debugging techniques and is therefore an armored virus. We need to modify our pattern file to detect this virus. (We still need to recognize viruses that use a tricky jump sequence). Here are the details.

1. Armored virus. As discussed in the lectures on encrypted viruses, a common technique is to use decryption to disguise the virus code. To make the analysis even more difficult, the virus writer uses the stack pointer (%esp) in the decryption routine (anti-debugging technique). We can use this fact to detect the virus. The sequence that the Wahoo virus uses begins with the following code (comments were added by the experienced MOM Aurors):

```
leal  Start, %esi   ; load start address of location of virus payload
movl  $1666, %esp   ; put length in stack pointer
xorl  %esi,  (%esi) ; begin decryption
xorl  %esp,  (%esi)
```

2. The MOM virus analysts have determined that a good signature for recognizing the Wahoo virus is:

```
movl  <con>, %esp   ; put length in stack pointer
xorl  %esi,  (%esi) ; begin decryption
xorl  %esp,  (%esi)
```

   where <con> is a constant in the range (including 1400 and 1876) 1400 through 1876 (the MOM Aurors have determined that this insidious virus also uses different payloads that have a range of sizes).

Modifying virus-patterns.l
1. Like the previous assignment, we will be using the filter program to write a stream of ASCII characters tostdout which will then be read by the flex-generated scanner.

2. Modify your flex pattern file to also detect the Wahoo virus

3. Do not write anything else to the output stream. A sample run and output would be:

```
$ ./filter <infected-target01.exe | ./scanner
WARNING! Tricky Jump: byte number: 1076
684c840408c3

WARNING! Wahoo virus: byte number: 1087
bc8206000031363126
```

4. As before, call your flex file virus-patterns.l. Run and test your solution. You might want to create some input files that contain sequences that the patterns should match and which also hae sequences that have close misses that your scanner should not match.

5. Note: we will be testing your code on a variety of inputs to see if you get false positives or false negatives.

Submission Guideline

1. **WARNING: YOUR SUBMISSION ***MUST*** FOLLOW THIS GUIDELINE. THERE WILL BE 40% PENALTY FOR THOSE WHO FAIL TO FOLLOW THIS GUIDELINE.**
2. Submit your filter.c and *virus_patterns.l* to collab. **THE FILENAME MUST BE filter.c and virus_patterns.l .**
3. Your program will be compiled with the following command:

```
$ flex virus_patterns.l
$ gcc -o scanner lex.yy.c -lfl
```

   You will receive 0 points if your program fails to compile with this command. Warning messages from the compilers are OK.
4. **Your filter program MUST read from stdio and output to stdout.**