Source File: `~/public_html/lab14.php`
Input: URL
Output: Standard Output
Value: 2

In C and C++ integer literals come in three guises: decimal, octal, and hexadecimal. Decimal literals are the most commonly used and look as you would expect them to:

$$0 \quad +1234 \quad -976 \quad 12345678901234567890$$

The compiler usually issues a warning about literals that are too long to represent. Integer literals may be preceded with an optional sign (either '`+`' or '`-`').

A literal starting with zero followed by `x` (`0x`) is a hexadecimal (base 16) number. The `x` may appear in either case. A literal starting with zero followed by a digit is an octal (base 8) number. For example:

decimal:	`0`	`+2`	`-63`	`83`
octal:	`00`	`+02`	`-077`	`0123`
hexadecimal:	`0x0`	`+0x2`	`-0x3f`	`0x53`

The letters `a`, `b`, `c`, `d`, `e`, and `f`, or their uppercase equivalents, are used to represent 10, 11, 12, 13, 14, and 15, respectively. Octal and hexadecimal notations are most useful for expressing bit patterns.

The suffix `U` (or `u`) can be used to write explicitly `unsigned` literals. Similarly, the suffix `L` (or `l`) can be used to write explicitly `long` literals. For example, `3` is an `int`, `3U` is an `unsigned int`, `3L` is a `long int`, and `3UL` is an `unsigned long int`. The suffixes may appear in either case and in any order.

The purpose of this lab is to write a program that will accept an integer and determine if the number is a valid or an invalid integer literal in C++. Use a regular expression to make this determination.

Your program will accept a string from the URL via which the program is accessed. Let this string represent the integer literal that is to be tested. The output should be formatted as shown in the instructor's version of this program.

Some additional notes for this assignment:

- Insert an HTML comment at the top of the document identifying you as the author, the class, and the assignment number.
- Add an echo statement to the beginning of the script section that will display your name, the course number, and the assignment number.
- Since this assignment uses several PHP code blocks, it's always a good idea to check for syntax errors. You can do this by using the `-l` option to the `php` command at the command line as in

```
1 newuser@csunix ~/public_html> php -l lab14.php
2 No syntax errors detected in lab14.php
```

- You should always validate the rendered HTML code. The validator is discussed near the top of p. 6 and in Appendix A on pp. 629–631. By including the following link and image, a user will be able to click the image and receive a report from the validator.

```
1 <?php
2   $location = 'https://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'];
3   $location = urlencode($location);
4   echo '<a href="https://validator.w3.org/nu/?doc=' . $location . '">';
5 ?>
6   <img src="https://www.w3.org/QA/Tools/I_heart_validator"
7    alt="I heart Validator logo" height="31" width="80" />
8 </a>
```

After the document is valid, open it in your Web browser to see how it renders.

Upon completion of this assignment, submit your source file via Blackboard.