CSci 130, Fall 2001, Exam 2

Name (please print): _____ ID #: _____
First Last

- This is a 50-minute closed-book exam. There are 4 questions, worth a total of 100 points.
- Please read **all** of the instructions carefully before beginning to work on a problem, and if anything is not completely clear, please raise your hand and ask.

1. _____
2. _____
3. _____
4. _____
Total. ☐

Question 1. (15 points) Define a Perl subroutine, called `greater`, that takes two arguments, an integer and a *reference* to an array of integers, and returns an *array* consisting of all of the elements from the argument array that are greater than or equal to the argument integer, in the same order as they appear in the argument array. For example, after the two statements

```
@a = (8, 2, 3, 9, 10, 1, 5, 2);
@b = &greater(5, \@a);
```

The array `@b` would be `(8, 9, 10, 5)`. Your subroutine should **not** introduce any global variables.

Question 2. (22 points) Define a Perl subroutine, `unique_lines`, that takes two filenames (strings) as arguments, and stores into the second file all of the lines that appear in the first file, in sorted order (using the default alphabetical ordering), with all duplicates removed. For example, if the file "`input.txt`" contained the four lines on the left below, then after executing the function call

```
&unique_lines("input.txt", "output.txt");
```

the file "`output.txt`" would contain the three lines on the right below:

```
efg abcd                    abc def
abc def        --           cd
efg abcd                    efg abcd
cd
```

(Hint: think about the "grades" example from class, and how student names were collected, sorted, and printed without duplicates in that example.) Your function must **not** introduce any global variables.

Question 3. (25 points) Write a CGI script in Perl, *without using the `CGI.pm` library*, that counts the number of keyword-value pairs submitted to it. In more detail, your script must work when requested with either the `GET` or `POST` method, and return an HTML document whose body consists of a single number, the number of keyword-value pairs submitted to it. For example, if the URL of your script is `http://www.host.com/count.cgi`, then both the `GET` request

```
GET http://www.host.com/count.cgi?a=1&b=2&c=3 HTTP/1.0
```

and the `POST` request

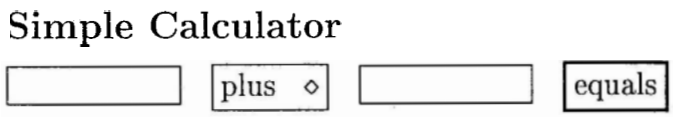
```
POST http://www.host.com/count.cgi HTTP/1.0
Content-Length: 11

a=1&b=2&c=3
```

would result in the HTML document

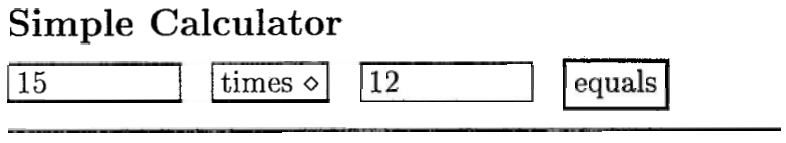
```
<html><body>3</body></html>
```

being returned (because there were 3 keyword-value pairs submitted).

Question 4. (38 points) Use the `CGI.pm` library to create the following CGI/Perl application. When visited for the first time, your script should produce a form that looks like



which consists of an H1 header with text "Simple Calculator", a textbox, a pull-down menu with options "plus" and "times", another textbox, and a submit button with text "equals". The title of this document is "plus/times". When the submit button is pressed, the form will be reprinted, with all data and selections intact (i.e., it is a "sticky" form), and then, below a **horizontal rule**, the numerical answer will be given. For example, if the user types 15 into the first textbox, selects "times" from the pull-down menu, types 12 into the second textbox, and clicks the "equals" button, then the result would be



All headers and HTML tags **must** be produced by method calls on your query object.