

# CS193i Final Exam

---

- **SITN Students:** in theory, this exam has been emailed or faxed to your coordinator so expertly that you are able to take the exam on our regular day: Fri, which is good since we're doing the grading Sat. Please fax it back as soon as you are done with it. -- fax to Nick Parlante, 650 723-6092. Make sure you only write on one side of each page, or the Fax won't pick it up. Or if that didn't work, you may take it on Mon and fax it in then.
  - This is a 3-hour, open book, open note, closed computer exam. There are 6 problems across 19 pages.
  - **Please write only on the front sides of pages** — if you need to add pages, please attach them to your exam with your name on them.
  - Things like syntax, exact method names, import statements, etc., are not important.
  - There is some time pressure, so if a problem is getting away from you, skip it until later.
  - For Sale: Parachute. Only used once, never opened, small stain.
- Good luck, and have a great summer!

Name (Last, First): \_\_\_\_\_  
(please try to print neatly!)

Please check here if you are graduating this quarter: \_\_\_\_

Please check here if you are a P/NC student: \_\_\_\_

I agree to the letter and spirit of the Stanford honor code — that no illicit aid has been given or received on this exam.

Signed: \_\_\_\_\_

1: \_\_\_\_/10 \_\_\_\_

2: \_\_\_\_/10 \_\_\_\_

3: \_\_\_\_/30 \_\_\_\_

4: \_\_\_\_/15 \_\_\_\_

5: \_\_\_\_/15 \_\_\_\_

6: \_\_\_\_/20 \_\_\_\_

\_\_\_\_/100

## 1. Socket (10)

For this problem, you will write "delivery" socket Perl code that picks up text from one host and delivers it to another. You may assume that all text lines use a simple "\n" end-line convention. Ignore all errors -- assume that all I/O operations succeed and that all the data is formatted correctly. To connect client sockets, assume that a Perl "conn" function is defined:  
 conn(SOCK, *host-name*, *port-num*).

- Open the file "hosts". Each line in the file contains a hostname and port number separated by a space. We'll call these the 'X' hosts...

```
aaa.foo.com 8080
bbb.bar.com 9090
...
```

- Make a socket connection to each X host on the given port number. It will send back a sequence of text lines, followed by a blank line.
- After the blank line, the X host will send one or more lines that identify hosts and port numbers with the same syntax as above. We'll call these the 'Y' hosts. Open a connection to each Y host, send it the text lines from the X host from before the blank line, and close the connection.

```
blah blah blah
yatta yatta
text text text
<blank line>
ccc.abc.com 123
www.bbb.edu 1000
```

Your Perl code here...

```
#!/usr/bin/perl
```

```
## more room
```

## 2. HTTP (10)

For this problem you will write some simple Java HTTP client code. You may omit all error handling and exception code. Complete the code for the method

```
public String webSearch(String host, String path, String target) {
```

- The host and path parameters represent the data from a url. For the url "http://foo.com/bar", the host parameter is "foo.com" and the path parameter is "/bar".
- Connect to port 80 on the given host (code provided below)
- Send an HTTP request (1.0 or 1.1) for the given path.
- Read back the HTTP response. (You may ignore the "200" result code -- proceed regardless of the result code)
- If the returned HTTP document body is a textual and it contains the target string somewhere in one of its lines (case-sensitive), then return the first matched line, otherwise return null.

```
    Socket sock = new Socket(host, 80);

    BufferedReader in = new BufferedReader(
        new InputStreamReader(sock.getInputStream()));

    PrintWriter out = new PrintWriter(sock.getOutputStream());

    // YOUR CODE HERE
```

```
// more room
```

### 3. CGI (30)

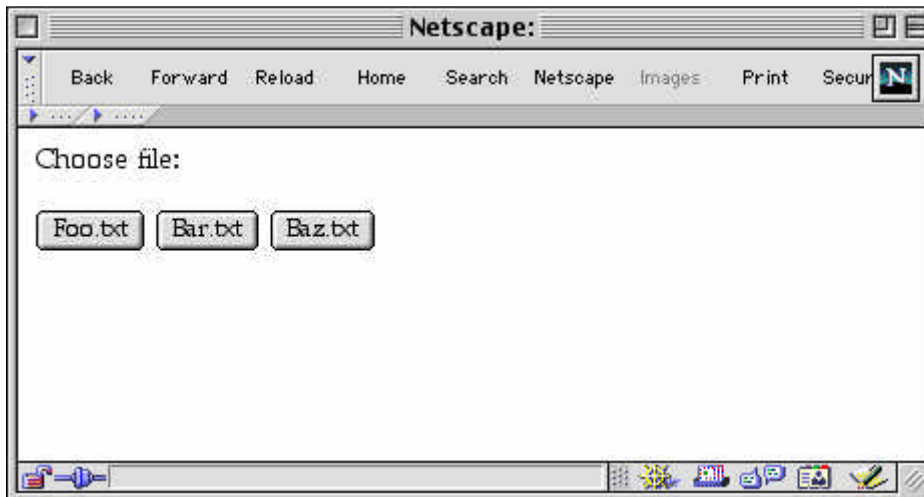
For this problem, you will implement a 3-page CGI that lets people edit their files on the CGI server. Here is the first page (P1), used to get the user name from the client side...



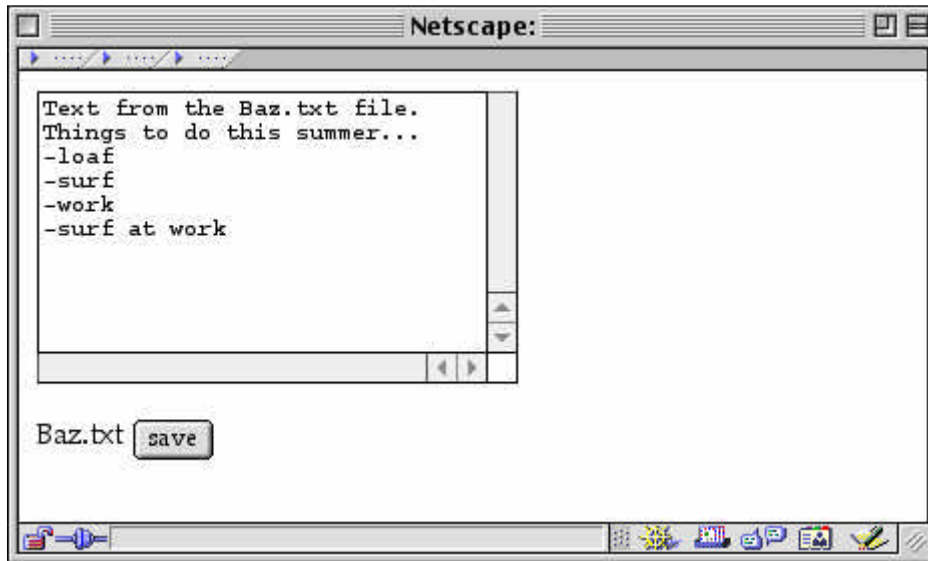
On the server side, there is a "users" text file. Each line in the file gives a username followed by the files they can edit, each separated by a space. The username and the filenames do not contain spaces.

```
bob Foo.txt resume.txt binky.txt  
alice jokes.txt Bar.txt  
.....
```

If the user submits P1 with a valid username they get P2 (below), otherwise they get P1 again. P2 lets the user choose from among the files they can edit...



Choosing a file on P2 leads to P3, where the user can see and edit the text contents from the given file (assume that the file contains plain text).



The "save" button saves the text back to the file, and returns to the P2 for this user. (The "text from the Baz...." is the first line in the file in this case.)

### Assumptions

- Ignore all error cases and do not worry about security.
- Do not worry about the back button -- the user must move forward.
- The Perl expression `split($str)` returns an array of the strings in `$str` that were separated by whitespace.
- Use forms to implement the solution, not cookies. (The same as HW3).

```
#!/usr/bin/perl  
  
use CGI;  
  
$q = new CGI;
```

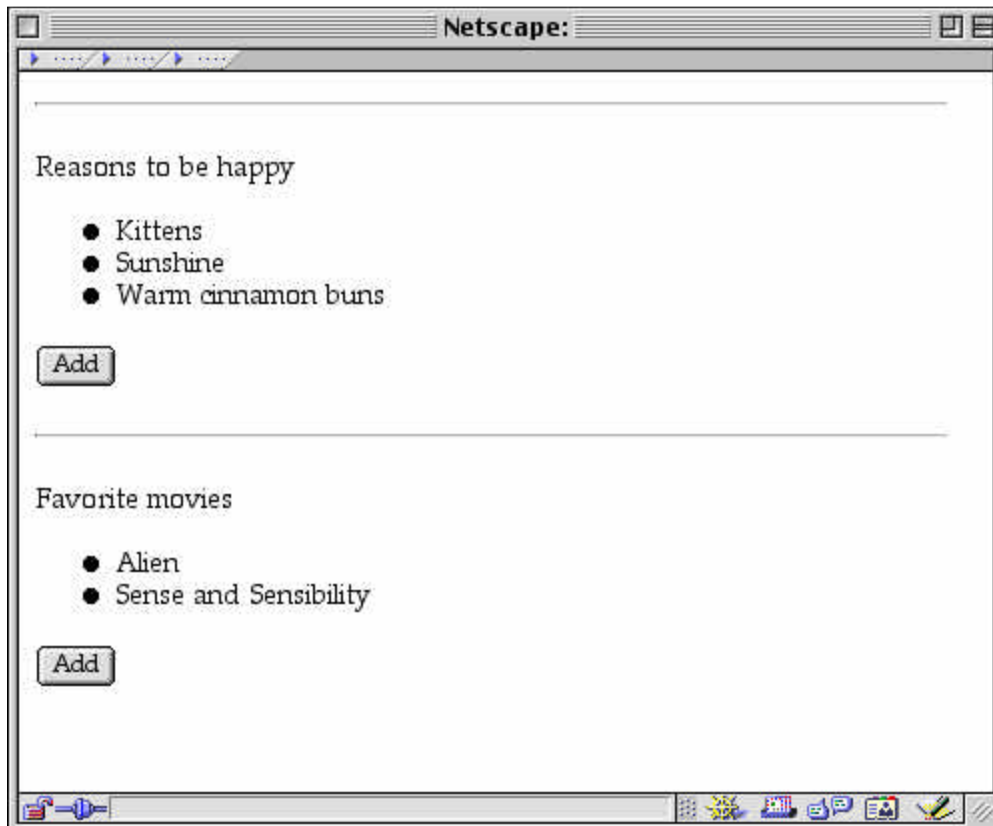
```
## more CGI
```

```
## even more CGI
```

## 4. Servlet (15)

For this problem, you will implement a Java Servlet that implements a trivial sort of weblog that presents categories where people can add their comments (slashdot.edu?). The basic features are simple, except we add the requirement that the servlet resist user errors with the browser Back button (detailed below).

Here is the first page (P1) the user sees. There are several categories ("Reasons to be happy", "Favorite movies"), and within each category, users can add their comments for all to see...

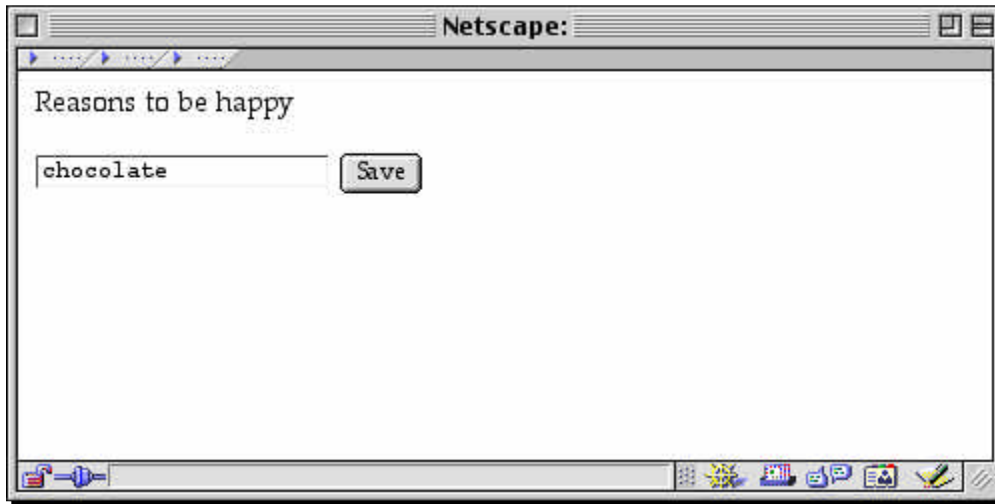


In the Servlet, the data is stored in a 'categories' instance variable that is an array of Category objects. Assume that the Category class is already defined, and responds to these messages...

```
public class Category {
    public String getTitle();           // get the title
    public String[] getComments();     // get the array of comments
    public void addComment(String comment); // add a comment
    ...
}
```

Your code can simply use the 'categories' array and the Category objects to access the data; Assume that the Category class deals with the back-end file or database storage. The number of categories does not change.

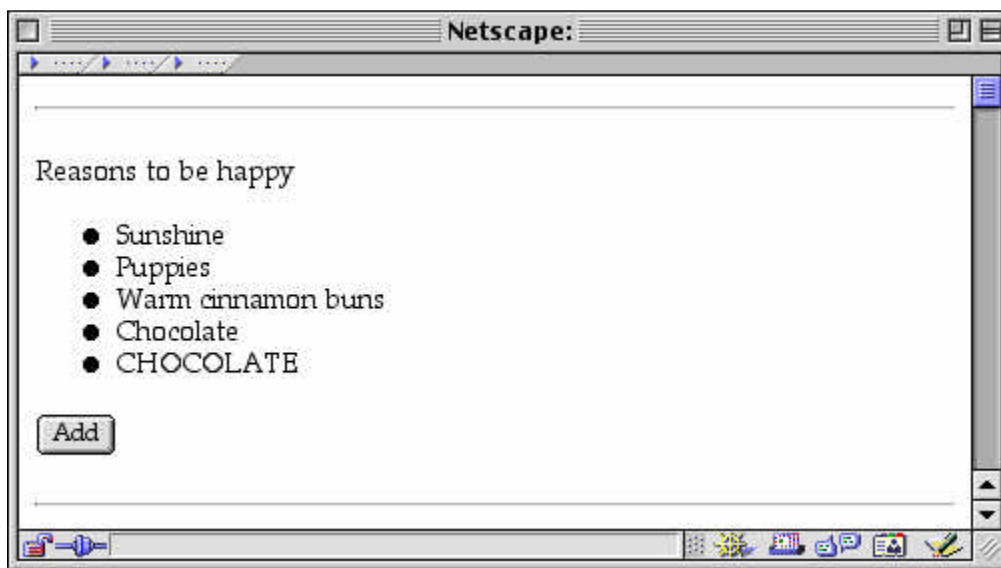
When the user clicks the Add button, they see page 2 (P2) for that category. It shows the category title and has a form for adding a comment...



Clicking the Save button adds the comment into the data for that category, and displays P1.

On top of this fairly simple structure, we will add one tricky requirement. The servlet must resist a specific user error due to the browser Back button....

- Suppose the user goes to P2, types in the comment "Chocolate", and clicks the Save button. This adds the comment and displays P1.
- Then the user clicks the browser Back button, to re-visit P2, changes the comment to "CHOCOLATE", and then clicks the Save button.
- Problem: with the obvious Servlet (or CGI) implementation, this sequence will cause two comments to be added, which is probably not what the user wanted. In fact the same problem occurs if the user just clicks the browser Reload button when viewing P1 after a save



Your servlet must be resistant to the following specific "double submit" scenario:

Suppose the user clicks the browser Back button some number of times so they see an old instance of P2, and then click Save. Such a Save should be recognized as a "double submit". In that case, the servlet should not add a new comment or change the data in any way; it should just display P1. Even if the user changes the comment string in some way, the double-submit should be detected.

To implement this feature, the servlet should put some identifying information in each P2 when it is generated, to detect if it is ever submitted a second time.

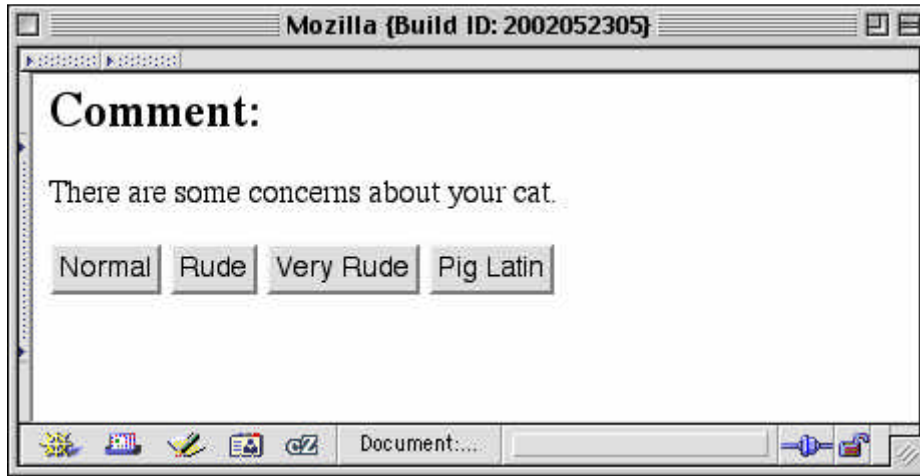
You may ignore servlet threading issues (we never covered that detail). Do not use cookies explicitly, but everything else (forms, ivars, sessions, ...) is ok.

```
public class Weblog extends HttpServlet {  
    private Category[] categories = ... ; // set up for you  
    public void doGet(HttpServletRequest request,  
                       HttpServletResponse response)  
        throws IOException, ServletException  
    {  
        // your code here  
    }  
}
```

```
// more space
```

## 5. JavaScript / Thick Client (15)

Suppose you are working on a servlet, and part of its output is a little "comments" section like this...



The servlet has different style versions of the same comment

<u>Style</u>	<u>Comment</u>
Normal	There are some concerns about your cat.
Rude	Your cat sucks.
Very Rude	Your cat sucks, and so do you.
...	

Suppose in the servlet, you have two arrays of identical length. The "comments" array contains the various style versions of the comment string, and the "style" array identifies the corresponding style ("Normal", "Rude", ...). The exact length and contents of the arrays vary at runtime.

**Objective:** When first displayed, the page should use the first comment in the array. The user should be able to click a button to change the comment style. For example, clicking the "Rude" button should change the comment to "Your cat sucks". In typical JavaScript fashion, this should happen without hitting the server. (There are quite a few workable JavaScript strategies to accomplish this -- anything that works is fine.)

Write the necessary fragment of servlet code in the `doGet()`, below, to produce the Comments section, and its buttons and JavaScript code. You may write part of your solution as a JSP if you wish. You do not have to write out the `RequestDispatcher.....include(...)` fragment. We'll assume it's there if needed. Assume that the variables `out`, `comments` and `styles` are defined...

```
public class Servlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        ...
        // assume standard servlet stuff here
        ...

        // Assume that PrintWriter out, String[] comments, and String[] styles
        // are defined.
        // YOUR CODE HERE -- produce the comments section
    }
}
```

```
// more space
```



Suppose Alice wants to do secure communication with Bob, but neither has a certificate, so they are vulnerable to an MITM attack. Alice sends the Alice.pub key to Bob so that he can securely send back a session key, k. Unfortunately, Carl is performing an MITM attack on all their communication...

g(1). What is the first thing that Bob receives?

g(2). Bob sends back a session key, Bob.k. What does Alice receive? (there are a couple possibilities -- identify one)

h. True or false: If the network administration tool for a machine reports that 'promiscuous mode is off', then we can conclude that the machine is not sniffing the network traffic of its neighbors.

i. Suppose that Bob has keys Bob.pub and Bob.priv, and Alice wishes to send a document to Bob securely. True or false: Alice encrypts the document with Bob.pub, sends the encrypted form to Bob, and he decrypts with Bob.pub.

j. True or false: A "warhol" worm spreads quickly because its inner "probe" loop is coded to send packets very quickly.

k. Suppose a NAT router has the IP address 1.2.3.4 on its WAN/Internet side, 192.168.1.1 on its LAN side, and there is a host on its LAN side with address 192.168.1.100. That host wants to communicate with 9.8.7.6 out on the Internet, and so sends out a packet of the form (From:192.168.1.100 To:9.8.7.6). True or false: The NAT router rewrites the outgoing packet to the form (From: 192.168.1.1 To: 9.8.7.6).