

CS193i Final Exam

- **SITN students** -- hopefully you are getting this soon enough to take it on Tue or Wed. I would like to get the exam back in my hands by Thu. If the completed exam is not going to make that deadline, then please fax it to 650-723-6092. If you also send back your physical exam, write "duplicate of faxed copy" on the physically sent copy so we don't grade it twice!
- This is a 2-hour, open book, open note exam. Your exam should have 15 pages. There are 4 problems, each with a few parts. **Write only on the front sides of pages** — if you need to add pages, I have brought blank paper and a stapler. Write your name on any extra pages.
- Syntax, exact method names, import statements, semi-colons, etc., are not important. Do not worry about public vs. private classes or which class is in which file. We are more interested in the core tech issues in each question.
- The answers are not intended to need to be especially long and there is some time pressure — if you find yourself doing something very complex and time consuming, you may want to skip that question since it is probably not your strongest area.
- Carpe Post Meridiem!

Name (Last, First): _____
(please print neatly!)

Status: graduating senior: check here
 p/nc student check here

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: ____/30 ____

2: ____/25 ____

3: ____/25 ____

4: ____/20 ____

____/100 total

Today's joke: Doctor to Patient: I'm sorry but I have some bad news for you. It looks like there's only Ten... Patient: Ten! Wad'ya mean Ten! You mean I've got only ten years, or ten months to live!?? Doctor: Nine!

1) Short Answer (30 points)

These 19 little questions have simple, short answers — no more than 6 words, and often just one word. These are worth 1 or 2 points if correct, 0 point if left blank, and -1 or -2 if answered incorrectly. The 2 points questions are marked with a "(2)"; all the others are 1 point. Your best strategy is to go through quickly and answer the ones you know immediately, and leave the others blank. Crossing out 100% of your answer is equivalent to leaving the question blank. Your overall score for this question cannot go below 0.

- a. True or false: "For IP communication with Ethernet, the Ethernet packet is transmitted inside of an IP datagram".

- b. True or false: the first router that gets a datagram computes the "route" -- the sequence of hops for that datagram to reach its final destination.

- c. True or False: An IP datagram whose final destination is a particular computer contains the IP address of that computer.

For these three questions, suppose the Perl variable `SOCK` has just been connected to port 80 on the `foo.com` HTTP server, and you are initiating an HTTP1.0 connection.

- d. (2) Write a print statement that will initiate the retrieval of the document `http://foo.com/doc/bar.html`

- e. (2) Write a print statement that will initiate the retrieval of the document `http://foo.com/cgi-bin/a.pl?pi=3`

- f. (2) Write a print statement that will initiate the retrieval of the document `http://foo.com/cgi-bin/a.pl#pi=3`

For these two questions, assume you are parsing HTML retrieved from the URL `http://foo.com/a/b.html`

- g. (2) What is the full URL for ``
- h. (2) What is the full URL for ``
- i. True or false: when using the `Location: redirect` in a header, the HTTP server includes the text of the new destination page in the body section after the header.
- j. (2) How does an HTTP client know that the HTTP server is sending back HTML data as opposed to, say, JPEG data?
- k. How is an HTTP cookie sent from the HTTP client to the HTTP server?
- l. (2) What are the three reasonable techniques that an HTTP server application can use to give the "session" illusion of continuity across a sequence of pages?
- m. (2) If a JSP sets a cookie on the response after having printed out all the HTML, the cookie does not work (this is why we do cookies before doing anything else in a JSP). Why?
- n. Why are servlets faster than CGI's (for the standard implementation of CGI)?
- o. (2) For this problem, suppose X is the superclass of Y. Is the assignment below a compile time error?

```
X x = new X();  
Y y = new Y();  
  
y = x;    // Is this a compile-time error?
```

p. (2) Once again, X is the superclass of Y. What is the output of the Y.demo() method for the code below?

```
class X {
    void foo() {
        System.out.println("X foo");
    }

    void bar() {
        System.out.println("X bar");
        foo();
    }
}

class Y extends X {
    void foo() {
        System.out.println("Y foo");
    }

    static demo() {
        Y y = new Y();
        y.bar();
    }
}
```

q. (2) Here is some HTML source....

```
<table border=1>
<tr>
<td>
<p> a
  b c d
</td>
<td>
<p> a b
  c d
</td>
</tr>
</table>
```

Please draw how the HTML table might render in the browser...

r. What is a "replay attack" and what is a simple way to guard against it?

s. What is one simple operation that a "firewall" does?

2. TCP/IP (25 points)

For this problem, you will write some client Perl code for the Binky protocol.

For this problem, a "binky id" is a hostname followed by a port number in angle brackets —"sunburn.stanford.edu<45>" or "yahoo.com<9000>". The port number and brackets may be omitted, in which case the port is assumed to be 100, so binky id "foo.com" would use port 100.

The paragraph below describes overall computation. Your solution will be divided into four little parts, and each part repeats the description of what it is supposed to do.

The file "binky.txt" contains binky id's, one per line. For each binky id in the file, do the following processing: connect to the host and port number and write "out\n" to the server. The server response will have two parts, (a) a series of "nested" ordinary DNS host names, one per line, followed by a blank line (simple "\n"), followed by (b) HTML text. For each of the nested host names, connect to the given host on port 999, send the request "in\n", and simply read and print out all that it sends without modification. Do this for each of the nested host names. Then print out the blank line and all of the HTML that follows it.

You do not need to include any error handling code -- assume all connections succeed. Also, you do not need to declare your variables or have the correct "use" clauses.

For reference, here is a simple subroutine that will make a socket connection. You may call it from your code, or copy code from inside it.

```
## Open a socket for a given host and port
## Call like this: simpleConnect(SOCK, "www.yahoo.com", 80);
sub simpleConnect {
    my($sock, $host, $port) = @_;

    my($ipaddr) = inet_aton($host);
    my($sockaddr) = sockaddr_in($port, $ipaddr);
    socket($sock, PF_INET, SOCK_STREAM, 0);
    connect($sock, $sockaddr);
}
```

a. Open the file "binky.txt", start the while loop to go through all of its lines, parse each binky id into the variables \$outhost and \$outport (leave the body of the while loop open so it contains parts (b) etc. below). Use 100 as the default port number. You may assume that the binky id's are properly formatted.

b. Given \$outhost and \$outport from above, make the connection and send the "out\n" request. Start a while loop that reads each nested hostname into the variable "\$inhost" which will be processed in the next step.

c. For each \$inhost, connect using port 999, send the "in\n" request, read and print out all that is sent back.

d. When done with the nested hostnames, read and print out the HTML that follows the nested hostnames. Loop back to (a) to continue processing the file.

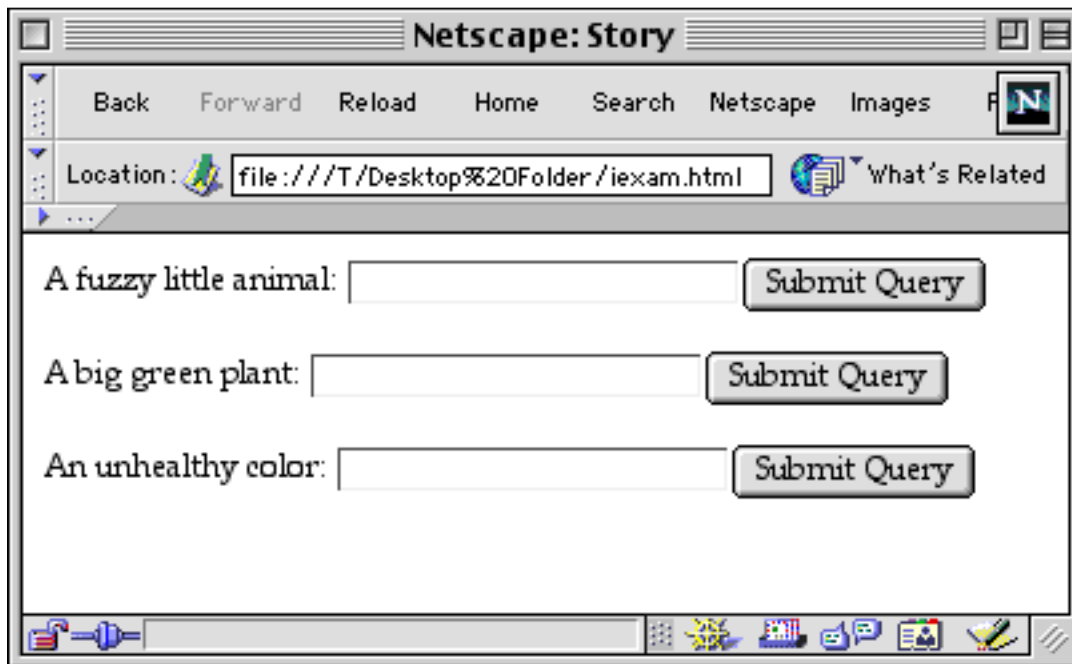
3. CGI (25 points)

For this problem, you will use the CGI Perl module to build a simple story-writing application. The file "story.txt" is divided into two parts separated by a blank line ("\n"). The top part is made of lines made of a single "name" word followed by a space, followed by a "description" phrase. The second part of the file is just regular HTML text, except some lines will contain only a single tag using one of the names from the first part...

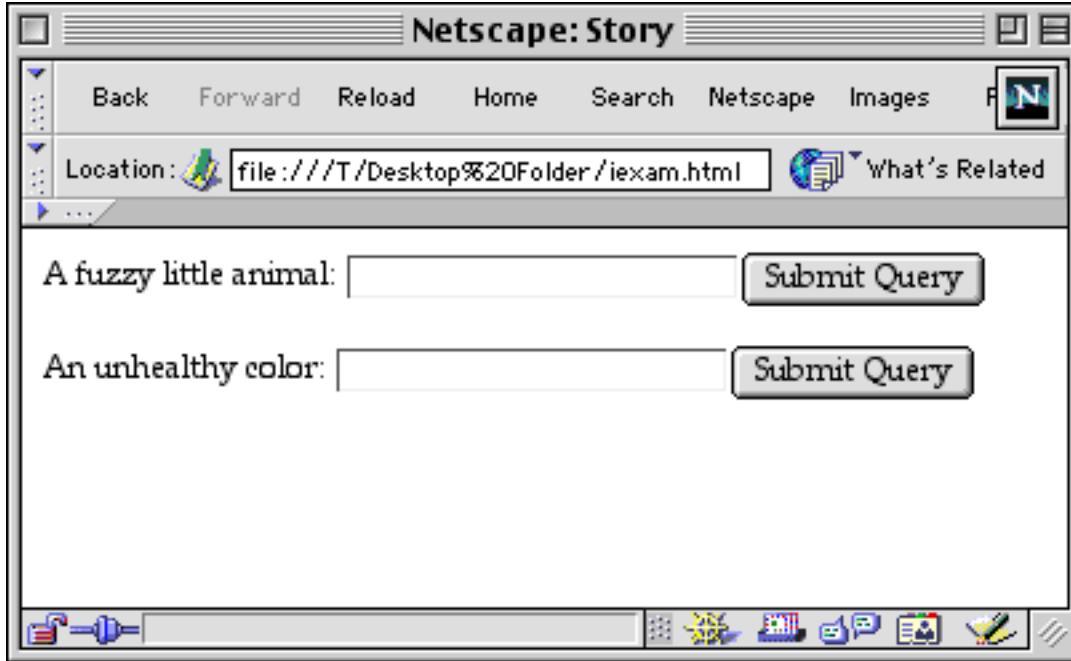
```
animal A fuzzy little animal
plant A big green plant
color An unhealthy color
```

```
<p>
Once upon a time
there was a cute little
<animal>
that enjoyed eating
<plant>
until it turned
<color>
```

When called with no bindings, the CGI produces one input field for each line in the first part of the file. The second part of the file is not used...



The user enters text in one of the fields and clicks the adjacent button to define the text for that name. This leads to a page with that input field no longer present. So entering "fennel" for "A big green plant" above results in the page...



The user will enter text for each field one at a time until all have been entered. The user may enter the fields in any order, however once one has been entered, the user does not get a chance to change it. We will not support the "back" button to go change old decisions — we will only go forward one entry at a time. Part (a) of the problem deals with generating the above pages as the user defines all the fields. Part (b) (below) deals with printing out the story once all the fields have been defined.

The routine header part of the CGI is given here — write your code to follow it. The CGI should re-read the "story.txt" file each time it is run, but should not change file.

```
#!/usr/bin/perl -w
use CGI;

$header = <<EOT;
<html><head>
<title>Story</title>
</head>
<body bgcolor=white>
EOT

$trailer = "</body></html>\n";

$| = 1;    ## set STDOUT to be unbuffered
```

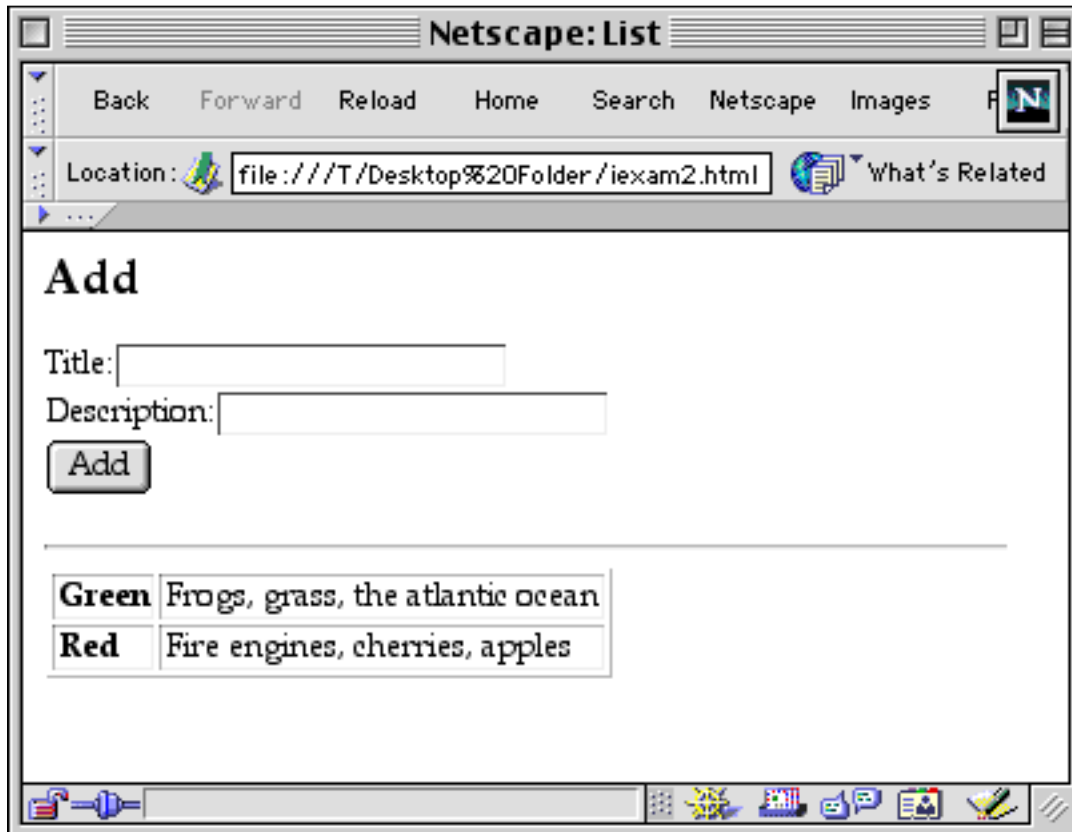
a. Set things up and go through the file "story.txt" and produce form entries for each of the names that has not yet been given any text by the user. If all of the names have been entered, then do not produce any form inputs and execute (b) below instead (in that case, it's ok to produce a form so long as it does not take up space on screen).

b. In the case that all the names have been defined, echo the HTML part of the file, using the text the user entered in place of each `<name>` tag. The name tags occur on lines by themselves in the HTML. The tags may appear in any order in the HTML and may each be used any number of times.

2. List Servlet (20 points)

For this problem, you will write a simple list servlet.

The top of the servlet output is always a little form which lets the user add a list item defined by a title and a description. The bottom part of the output shows the current list in a table. The title for each row is in bold. The list is maintained in a server side session, so if the user has two browser windows, they will manipulate the same underlying list. This is a very simple list application — items can be added, but they cannot be deleted or edited. Here's a screenshot after two entries have been added to the list...



There's an additional constraint that the lines inside the table must be generated using a JSP. The JSP will generate the HTML starting with `<tr>` and ending with `</tr>`. The servlet generates everything else. To help with the JSP, you have the `ItemBean` class which represents the "title" and "description" strings in the standard bean way...

```
// stores "title" and "description"
public class ItemBean {
    private String title;
    private String description;

    public ItemBean(String aTitle, String aDescription) {
        title = aTitle;
        description = aDescription;
    }

    public String getTitle() {
        return(title);
    }

    public String getDescription() {
        return(description);
    }
}
```

Here is the standard header for a Servlet — write your code to follow it...

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ListServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {

        // your code starts on the next page
```

a. Print out HTML and the "add" form. Locate the session object, notice if the add button has been clicked and if so change the session.

b. Print out the table, using "item.jsp" (part c) for each row

c. Define "item.jsp"