

Exam Practice

The final exam is the last two hours of our regularly scheduled exam time: Tue June 6th 9:30-11:30 a.m. in Cubberly Aud with overflow going to Annenberg Aud. There will be an alternate sitting of the exam Mon night — email Nick with the subject "alternate final" to get on the list for the alternate.

SITN students: you have the two regular choices: 1) You are welcome to join us in person for the regular reading of the exam. This provides the most authentic and traditional stressed-college-student-followed-by-catharsis final exam experience. 2) You may wait at your site, and the your coordinator can administer the exam there on Tues. I need to have the exam back in my hand (by courier or Fax) by Thu for grading. The coordinators should know how to deal with all this.

The final exam will cover all of the material we have seen this quarter. Most of the exam will be concerned with coding problems similar to the homeworks. A few of the problems will be essay/short-answer type on material from lecture. The short answer questions will be right=+2, blank=+1, wrong=+0, so it will not be in your interest to guess if you don't know the answer.

The exam will be 2 hours long and will be open book, open note, etc. The exam will be short enough (around 5 questions) that you will have sufficient time to quickly review and retrieve details from your notes, but there will not be sufficient time to *learn* the material during the exam. To prepare for the exam, you should be familiar with all of the major technologies and issues, and plan to rely on your notes and solutions for details during the exam.

1. Understand all the example code from in lecture.
2. Understand your own solution to each homework. The exam will include smaller but related code-writing problems. I will provide ".h" type information where necessary.
3. Be conversant with the major technologies, issues, and examples presented in lecture.

For code writing problems, we will not be especially picky about syntax or other basically conceptually shallow ideas. Instead, the questions will focus on understanding the key issues for each of the technologies we have covered: TCP/IP, HTML, HTTP, CGI (and to a lesser extent Perl), Java/OOP, servlets, and JSPs.

How To Study For A Code Writing Exam

A good way to study for the code writing questions is to take a coding problem for which you have a solution available (lecture example, homework problem fragment, sample exam problem) and try to write the solution as you would on an exam starting with a blank sheet of paper. Working through writing the code, even with errors, will give you a much better understanding of the major concepts than a *passive* review of the solution code. I am providing a large number of practice problems to solidify your understanding of the material — take advantage of the opportunity.

The rest of this handout includes a patchwork of questions from the last few years of exams — I've selected questions which are most similar to this year's coverage. See our web page for a complete repository of old exams (but the older they are, the less relevant they are to our quarter.) Our exam will look a little different since we have covered somewhat different material, but this should give you a good idea of what a code writing exam looks like. On the real exam, each question will be on its own page so there will be room for you to write your solutions directly on the exam paper.

4) Java/OOP Short Answer

a) X is a subclass of Y. Is the last assignment below a compile time error?

No, the assignment is not a CT error. (We also allowed the equivalent response "yes" if you wrote it right next to the "valid?" question)

```
X x = new X;
Y y = new Y;

y = x;    // valid?
```

b) What is the output of a call to B.demo()? (we are ignoring public/private issues here)

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

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

class B extends A {
    void foo() {
        System.out.println("B foo");
        super.foo();
    }

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

    static demo() {
        A a = new A();
        B b = new B();
        a.foo();
        b.foo();
    }
}
```

```
A foo
A bar
B foo
```

```
A foo
B bar    -- this most important line depends entirely on the axiom
         -- "the receiver never forgets its class"
```

d) Which generally has **more** properties— the superclass or the subclass?

The subclass.

1) TCP/IP Coding (20 points)

(Note: this question is from a quarter when we used C more — our questions will be language neutral between C and Perl.)

As background, here is the interface to the standard socket read() as it appears on our socket handout...

```
int read(int sock, char* buff, int buffLen);
```

Read bytes from a socket into the given buffer. Returns the number of bytes read, or returns -1 and sets `errno` on error. A return of 0 is an effective EOF— there are no more bytes. `read()` will not necessarily fill the entire buffer every time it is called. Typically, many successive calls to `read()` will be required to get all the bytes, either because they will not all fit in the given buffer, or because not all the bytes have arrived the first time `read()` is called.

One problem with the standard socket `read()` is that it is not convenient for reading lines of text. Ordinarily, each call to `read()` returns some unpredictable numbers of characters — possibly a fraction of line, or possibly several lines lumped together. For this problem you will write a `read_line()` function specialized to read text data one line at a time from a socket.

`read_line()` is like `read()`, except it tries to return a buffer containing exactly one line of characters up to and including a `\n` from the socket. Like `read()`, `read_line()` returns the number of bytes read, or 0 if there are no more bytes, or -1 if there is a socket error. As an additional error case, `read_line()` returns -2, if `buffLen` characters have been read into the buffer, but without yet seeing a `\n`. Like `read()`, `read_line()` should not null terminate the buffer. The simplest implementation strategy is to call `read()` repeatedly to retrieve 1 character at a time.

```
int read_line(int sock, char* buff, int buffLen) {
```

Answer. There were many ways to get a fully functional `read_line` but here's one correct example:

```
int read_line(int sock, char* buff, int buffLen) {
    int i, result;
    char ch; //so you don't need memory allocation

    for(i=0; i<buffLen; i++)
    {
        result = read(sock, &ch, 1); //&ch to get a char*

        if(result<=0) return result;
```

```

    buff[i]=ch;
    if(ch=='\n') return i+1;        //i+1 to account for the '\n'
}
return -2;
}

```

A lot of people decide to not take Nick's advice and to try to have a single call to read that attempted to read a full buffLen and then find a '\n' within the buffer. However, this solution is only appropriate if the characters beyond the '\n' are saved off in a global variable in order to be integrated into the next call to read_line. Such "over-reading" of the socket was penalized with a -10.

Our grading criteria was divided into 2 parts : 8 points for return handling and 12 points for read loop correctness. Some common problems with the read loop involved several potential misunderstandings of the behavior of read. Such problems included passing a char to read or a char * with no allocated memory (-2). A lot of people also assumed that read returned a '\0' terminated string (required for functions such as strcat) (-4/-6). Finally there were also several occurrences read_lines that attempted to manipulate the characters in a local buffer of size buffLen and then simply set buff to point to that local buffer not realizing that only the local copy of buff is affected(-4).

There were 4 return cases to handle (encountered '\n', buffer full, no more character to read, socket error) so each was assigned 2 points where one had to both detect and return correctly in order to get the credit. In addition, any major C error was assigned a small penalty (-1/-2) and minor C errors were just ignored).

2) Miscellaneous **Short** Answer. 1 word or 1 sentence answers— don't bog down with partial-credit explanations. If you know it great, otherwise leave it blank. There are 16 sub-parts. (25 points)

a) What is the difference between GET and HEAD in HTTP?

GET returns the HTTP header followed by the document. HEAD only returns the HTTP header.

b) Each of the following URL's is href'd in the page <http://hell.org/doc/punish.html>

For each URL, give its absolute form to the right...

- i) /binky.html
- ii) ironic/homer.html#doh
- iii) donut.gif
- iv) /test.html#contents

Key points: understand how relative URLs work, realize that #doh is part of the URL.

i) <http://hell.org/binky.html>

ii) <http://hell.org/doc/ironic/homer.html#doh>

iii) `http://hell.org/doc/donut.gif`

iv) `http://hell.org/test.html#contents`

c) What is the complete, exact string which the HTTP client sends to the HTTP server to request page (iv) above? Give your answer as a double quoted (") string constant as you would in C or Perl...

`"GET /test.html HTTP/1.0\r\n\r\n"`

Key points: remember how to put together an HTTP request which SiteSucker covered rather extensively.

d) How does an HTTP client know the type of the data returned by the server?

The Content-Type: field in the HTTP header returned by the server.

e) 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?
```

Yes, this is a compile time error. Subclasses have the properties of their superclasses and can therefore stand in for their superclasses. Not the reverse.

f) X is the superclass of Y. Both classes define a void foo() method. On the "x.foo()" message send -- which method definition is invoked: the X class' or the Y class'?

```
X x = new Y;

x.foo(); // which method does this invoke?
```

Invokes the y.foo() method. This is the "objects always remember their class" rule in action.

g) Once again, X is the superclass of Y. What is the output of the Y.demo() method?

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

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

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

```

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

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

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

```

Y Test

X Test

Y Foo (this is the most important line)

X Bar

i) Most of our current Internet technology is based around public standards which have grown to have high quality implementations. Historically, many proprietary standards have not been widely adopted or well implemented. Why do public standards tend to eventually have better implementations?

Competition among multiple implementations. The marketplace of multiple implementors drives the quality of the implementations available up. There's a related virtuous cycle — because people expect the public standard to be better supported, they support it.

j) When you give your browser a URL such as "http://www-cse.stanford.edu/" — most likely what host must the browser first contact as part of retrieving the page (somewhat tricky).

The local DNS server to look up the IP addr of www-cse.stanford.edu. One person cleverly pointed out that technically, the most likely computer contacted will be the first router on the path towards the local DNS server, although we could dispute whether the router counts as a "host".

k) What are the two main things encoded in an IP address?

The host # and its network #.

l) Bob Metcalfe and Larry Wall are two very influential people for the content that has become CS193i. Which one of them invented Ethernet?

Bob Metcalfe. A little additional research suggests that there's an 'e' at the end of Metcalfe, although both spellings are used widely on the net!

m) What prevents IP datagrams from getting stuck in "infinite loops" in the Internet — forwarded from X, to Y, to X, to Y...

The Time-To-Live counter in each datagram. "Routers" or "TCP/IP" were not sufficiently specific.

n) True or False. When an IP router receives an IP datagram, it looks at the IP address of the final destination and figures out all the routers the datagram will need to traverse on its way.

False. It is largely concerned just with the "next hop".

o) Here is some HTML source.

```
<p>This dress <p>exacerbates the genetic
betrayal
that is <p> my legacy
```

Please draw how the HTML might render in a browser window which is..

| <----- this wide -----> |

This dress
exacerbates the genetic
betrayal that is
my legacy

Key points: realize that whitespace in the HTML source is ignored. Wrap the HTML appearance to the browser width, and put in vertical whitespace (wt) between the three paragraphs. The quote is from the embittered Janeane Garafolo in "Romy and Michele's High School Reunion "

2) Miscellaneous Short Answer (1 or 2 sentence answers)

a) With the current IP scheme, you cannot move a computer, say, across campus, plug it into a different physical net, and expect it to work. What information is encoded in an IP address and why does that scheme make it difficult to move computers around?

The IP address is broken into "net" and "host" numbers. The net number does not necessarily correspond exactly to a physical net, but it is related since it is a function of the overall topology of the Internet. Therefore, moving a computer from one physical net to another will very likely necessitate that it be "in" a different logical IP net number. Routers use the "net" part of an IP address to guide their routing decisions in real time. This is a basic tradeoff in the IP addressing scheme— encoding net numbers in the IP addresses makes routing easier, but it makes the IP addresses less flexible, and needing a new IP address when your computer changes nets is one example.

bc) CGI is a standard. Roughly speaking, what process does it control and what 3 software entities (programs) are involved?

CGI (Common Gateway Interface) mediates the dialog between three programs: an HTTP client, an HTTP server, and a CGI compliant program on the server. Typically the client sends a request, the HTTP server fields the request, maps the request to the CGI script, and sends the output (probably HTML) back to the client.

d) Name an advantage of client/server computing over the mainframe/terminal model.

Many possible answers. Many people quoted from lecture "computing power at client end allows a more responsive / facile interface". "Can potentially exploit computing power at client end." "Allows efficient division of problem between client and server." Marginal answers included "less network bandwidth required" or "allows centralized resources" since neither of these contrasts with the mainframe/terminal model.

e) Name an advantage of client/server computing over the application+documents-all-on-the-local-machine model (e.g. Macintosh, Windows...).

Many possible answers. The most popular answer (from lecture somewhere no doubt) "allow centralization of things which should be centralized." Most answers included phrases like "centralization" "avoid duplication" "avoid synchronization problems" "don't have local copies of things" "exploit expensive resources available only on the server".

3) CGI Coding

For this problem, you want to write a CGI script which allows users to pick a set of pictures to see. The CGI Perl code will use the same structure and the 193icgi.pl library as the homework.

You only need to write the "middle" part of the script which deals with generating and responding to the form. The context is..

- 1) (As on the homework) The %pairs associative array has been built by the 193icgi.pl library and the "content type: text/html" has already been written. Your code needs to write the entire HTML response. You do not need to deal with error cases gracefully.
- 2) There is an array called @names which contains the human readable names of a bunch of image files. e.g. ("Mona Lisa", "Hans Gruber", "Guernica", ...
- 3) There is an array called @urls, the same length as @names, which contains the corresponding URL for each of the images in @names.

a) Form case. For the case that the script is invoked without arguments, write the code which will generate a form page with the title "Image Picker" containing a single submit

button and an unordered list containing a checkbox for each image in the @names array. Each list entry should look like...

```
<li><input name = "Mona Lisa" type = checkbox>Mona Lisa

# keep HTML header and trailer in vars (used below)
$html_head = << END_TEXT;
<html>
<head><title>Image Picker</title></head>
<body><h1>Image Picker</h1>
END_TEXT

$html_tail = "</body></html>";

*** Errors...
*** not having <form ...></form>
*** not having <input type=submit>
*** not having <ul></ul>

# Here the form HTML is in a var, but you could do it in print stmts

$form_head = << END_TEXT;
<form method=post> # as on the HW, could have URL or not
Please pick images to view:
<p>
<ul>
END_TEXT

$form_tail = << END_TEXT;
</ul>
<input type=submit>
</form>
END_TEXT

#### The main part ####
# "if" can be assumed...
if(scalar(%pairs) == 0) {
    # Invoked with no arguments. Return a form.
    print $html_head;
    print $form_head;

    *** Some people tried to put this loop inside a string!
    foreach $name (@names) {
        print "<li><input name = \"$name\" type = checkbox>$name\n";
    }
    print $form_tail;
    print $html_tail;
    exit(0);
}
```

b) Response case. Write the code which triggers when the part (a) form is submitted by the user. Each checked checkbox will create an entry in the %pairs associative array with the name as the key and the value "on". Unchecked checkboxes do not add anything in the associative array. Generate an HTML document titled "Selected Images" which

embeds all of the checked images using the tag. Use the values in the @urls array to get the url for each image. The HTML for each image should look like...

```

<p>
<img source-specifier><br>
image-name

# $html_tail is from part a).

$html_head2 = << END_TEXT;
<html>
<head><title>Selected Images</title></head>
<body><h1>Selected Images</h1>
END_TEXT

# This is the essentially "else" part of part a).

print $html_head2;

# There is more than one way to do this.
# An alternate way is to loop through the %pairs, but you will need a
# way to find the correct index into @names and @urls.
foreach $i (0..scalar(@names)-1) {# (or could write as a while loop)
    if ($pairs{$names[$i]} eq "on") {
        # or if($pairs{$names[$i]}) {
        # or if(defined($pairs{$names[$i]})) {
        # or if($pairs{$names[$i]} =~ /^on$/i) {

        *** Errors...
        *** not having "src="
        *** assuming @urls is an associative array
        *** not considering that items in %pairs might not be in the
        # same order as in @names or @urls.
        print "<p>\n<img src=$urls[$i]><br>\n$names[$i]\n";
    }
}

print $html_tail;

exit(0);
}
}

```

1) TCP/IP and HTTP (25 points)

First, here is some reminder code for the socket interfaces you may want to consult for this problem. You do not need to check the error returns for reading and writing — just for hostname lookup and connecting.

C Socket Code

```

struct hostent *hostInfo;
struct sockaddr_in inetAddr;

```

```

int sock;

inetAddr.sin_family = AF_INET;
inetAddr.sin_port = <port num>;
hostInfo = gethostbyname(<hostname>); // returns NULL on err
memcpy(&inetAddr.sin_addr, hostInfo->h_addr, hostInfo->h_length);
sock = socket(PF_INET, SOCK_STREAM, 0);
connect(sock, &inetAddr, sizeof(inetAddr)); // returns -1 on err

once the socket is setup, you can read and write with...

int write(int sock, char* buff, int buffLen); // you may ignore the return value
int read(int sock, char* buff, int buffLen); // returns 0 on EOF

```

Perl Socket Code

```

$ipaddr = inet_aton(<hostname>); # returns false on error
$sockaddr = sockaddr_in(<port num> , $ipaddr);
socket(SOCK, PF_INET, SOCK_STREAM, 0);
connect(SOCK, $sockaddr); # returns false on error

```

Once connected, the socket can be read with <SOCK> and written to with:

```
print SOCK <string>;
```

For this problem, you need to write a small program which retrieves a series of web pages. Standard input (STDIN) to the program will consist of a series of absolute (not relative) HTTP URL with the leading "http://" omitted. Each URL ends with a complete path+file beginning with a single '/'. For simplicity, the path+file will not be the empty string and will not have a suffix. So the input will look like...

```

cse.stanford.edu/class/
www.whitehouse.gov/pets/socks.html
www.yahoo.com/

```

The program should print each URL to standard output and then try to do a single HTTP GET for the URL. If the hostname is bad or the connection fails, then print the error message "ERROR hostname" or "ERROR connection" after the URL with no further processing required for that URL. These are the only two error cases which the program must consider. Otherwise, if the connection is successful, the program should read all of the header and body of the HTTP response and simply print it all out as returned by the server. In any case, the program should then continue processing URLs until the input is exhausted. You may build your solution in C or Perl (just as on the homework), but you should not use any HTTP specialized libraries other than Socket. You do not need to worry about decomposition or style of your solution — we will grade only on the proper functioning of your code.

Solution notes:

Read each line from standard input with <STDIN>

Parse each line with a regular expression such as `m"(.?)/(.*)" == $1 is the host, $2 is the path`

Try to connect to the host on port 80

Send the GET request as "GET \$path HTTP/1.0\r\n\r\n"

Use the standard while (defined(\$line = <SOCK>)) { ... to read all the lines returned from the server

4) Servlet Programming (20 points)

(Note: the quarter this was given, the HW involved reading from a file. In a quarter where students did not do file reading on a homework, I would not ask for file reading on the exam).

For this problem, you will write a servlet that displays the text of books, where the user can censor out bad words. The servlet returns a page with three parts...

1) List Of Books

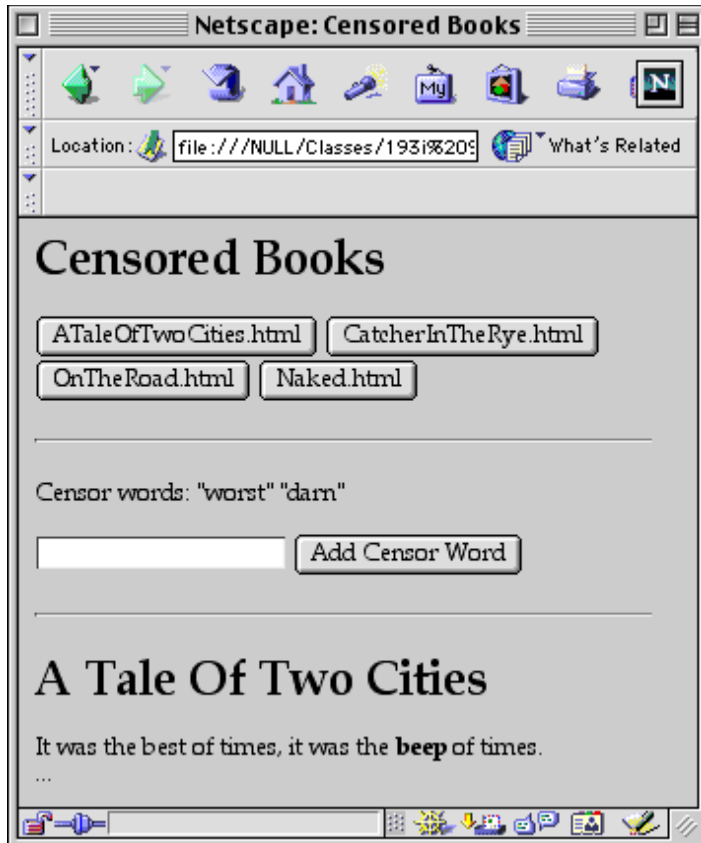
The top part of the page displays buttons, each of which can be used to choose a book. There's a Vector (below) that contains the book filenames which are used as the button titles.

2) Censor Section

The servlet keeps a list of words that it censors from the text of the books. The Censor section shows the words currently in the censor list. There is a text field and an Add button that allow the user to add a word to the list. Adding a word should return the same page, but with the word added to the list. Words can be added to the censor list, but they cannot be removed. Changing from one book to another, does not change the list of censored words. The list of censor words is used for the duration of the user's interaction with the servlet. Adding a censor word should not change which book is displayed in section (3) below.

3) Censored Book

If a book has been chosen, its HTML is pulled from its file and shown in the bottom of the page, except, any occurrence of a censored word in the HTML is replaced with a **beep**. Assume the file contains valid HTML that can be spliced right into the servlet HTML output. In this example, the words "worst" and "darn" have previously been added to the censor list. The servlet is currently showing the text of the book ATaleOfTwoCities.html. The opening sentence of this book begins "It was the best of times, it was the worst of times, ...". The word "worst" has been changed to **beep** since it is on the censor list.



Code

You may assume that the following methods are already defined in your servlet for your convenience. Your code does not need to deal with exceptions or try/catch blocks at all (if you know what those are). Your code does not need to recognize or deal with any error conditions.

- 1) Vector `getBooks()` returns a Vector of Strings of the file names of the available books.
- 2) `FileReader openFile(String fileName)` returns an opened `FileReader` object for the given filename. We will assume this operation always succeeds.
- 3) This code will read each line of a file as a String...

```
String line;
FileReader reader = openFile(fileName);
while ((line = reader.readLine()) != null) {
    // do something with line
}
```

- 4) Assume that the following message has been defined: `String superReplace(String s, String old, String replacement)` that replaces all occurrences of the substring **old** in **s** with the substring **replacement**, and returns the new resulting String.

```

// Censor

// (assume the import directives are all done for you)

public class Censor extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

// a) Generate the top book buttons (leave the session for part b)

        out.println("<html><head><title>Censored Books</title></head><body>");
        Vector files = getBooks();
        int i;
        out.println("<form>");
        for (i=0; i<files.size(); i++) {
            String fname = (String) files.elementAt(i);
            out.println("<input type=submit name=book value=\"\" + fname + \"\"");
        }
        out.println("</form>");

// b) Deal with the session object. Detect if they are adding a word
// to the censor list and update the list

        Vector censor;

        HttpSession session = request.getSession(false);

        if (session==null) {
            censor = new Vector();
            session = request.getSession(true);
            session.setAttribute("censor", censor);
        }
        else {
            censor = (Vector) session.getAttribute("censor");
        }

        String word = request.getParameter("word");
        if (word!=null and !word.equals("")) {
            censor.addElement(word);
        }
    }
}

```

```
// c) Echo the current state of the censor list and generate the
// form for adding censor words
```

```
out.println("<p>Censor words...");
for (i=0; i<censor.size(); i++) {
    String s = (String)censor.elementAt(i);
    out.println("\\" + s + "\\");
}
```

```
out.println("<p><form><input type=text name=word>");
out.println("<input type=submit value=\"Add Censor Word\"");
```

```
// d) If a book has been chosen, open its file, echo the HTML
// in the file, doing the censor replacement
```

```
String book = request.getParameter("book");
if (book!=null && !book.equals("")) {
    FileReader reader = openFile(book);
    String line;
    while ((line = reader.readLine()) != null) {
        int j;
        for (j=0; j<censor.size(); j++) {
            String cWord = (String)censor.elementAt(j);
            line = superReplace(line, cWord, "<b>beep</b>");
        }
        out.println(line);
    }
}
```