

Final Examination

Thursday December 6, 3pm to 6pm

Your name:

Instructions: Look through the whole exam and answer the questions that you find easiest first. Answer each question in the space below the question, using the backs of the pages for extra space as necessary. There are five questions, seven pages, and 110 points in total. Whenever you make an assumption, state it clearly.

You may bring and use the following materials: one PHP book and one MySQL book, your own personal hand-written notes, documents handed out in class, and a printed copy of web pages published for 134A, and web pages with direct links from 134A pages. You may not use any other materials.

(Question 1) [15 points]

(a) [5 points] Explain carefully how the following three statements are true and do not contradict each other: XML is not a programming language, VoiceXML is XML, and VoiceXML is a programming language. (Your answer should be about one paragraph.)

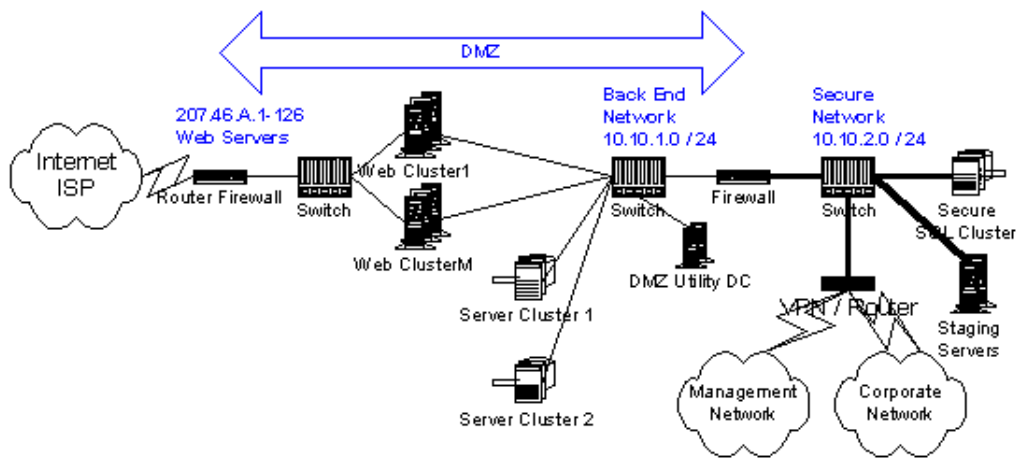
(b) [10 points] How can you maximize the usability of a VoiceXML application? Based on your experience with the VoiceXML project in 134A and elsewhere, explain what you think are the most important VoiceXML usability guidelines. (Your answer should be about two or three paragraphs.)

(Question 2) [15 points]

(a) [5 points] In the last project, you built a SOAP server with the same functionality as a U.S.P.S. server that does not use SOAP. Explain why the U.S.P.S. is likely to update its server to use SOAP in the future.

(b) [5 points] In the last project, you wrote XSLT stylesheets to generate ascii and HTML versions of business letters. Explain why many software experts recommend using XSLT instead of other languages for tasks like these.

(c) [5 points] Explain the meaning of the phrase “embrace, extend, dominate” in the software industry.



(Question 3) [25 points] This question refers to the figure above, taken from *A Blueprint for Building Web Sites Using the Microsoft Windows DNA Platform* published by Microsoft. Each answer to the parts below should include a short explanation.

- (a) [5 points] What does the acronym DMZ stand for? What is the DMZ?
- (b) [5 points] Why are there two firewalls? What does each firewall do? Should the router that connects to the corporate network be a firewall also? Why or why not?
- (c) [5 points] Assume that each web cluster is stateless, while each server cluster is a relational database system. For which type of cluster is it more difficult to achieve fault tolerance? Why?
- (d) [5 points] For each of the following types of data, explain where it should be stored. Justify each answer in one sentence.

1. The session identifier of a client.
2. Session data for a client, e.g. shopping cart contents.
3. Sensitive data for a client, e.g. home telephone number.
4. Public data such as product descriptions.
5. Log data for the web servers.

(e) [5 points] This architecture for a web site has at least one “single point of failure.” Identify this point and explain briefly.

(Question 4) [20 points] This question concerns basic PHP programming.

The task is to write a function named `stems` that finds all the word-stems in a sentence. A word-stem is the “root” of a word. To simplify, assume that for every word its stem is found by removing “s,” “ed,” and “ing” from the end of the word.

The sentence is given to the function as a string, and the stems must be returned as an array. In the array, stems must be lower-case, unique, and in alphabetical order. All punctuation must be removed.

For example, the result of `$a = stems("Democrats find Ken Starr's findings dubious.")` should be `$a == ("democrat", "dubious", "find", "ken", "starr")`.

(a) [5 points] Your function must be efficient in time and space. Explain in big-O notation what time and space complexity your function should have.

(b) [5 points] Your function should be coded in a modular way. Explain one to three reasonable “helper” functions you should use.

(c) [10 points] Consistent with your answer to parts (a) and (b), write PHP code for the `stems` function. Your code must be easy to understand, and must use appropriate built-in PHP functions.

(Question 5) [35 points] This question is based on the following example of session management adapted from the document *Session Handling with PHP4* by Tobias Ratschiller.

The following is completely commented sample code. The popular hangman game is a great way to show how to use persistent variables. In this game, the computer chooses a random five-letter word and the player needs to figure it out by choosing letters to fill in. The user has only six guesses, or he'll end up on the gallows. You're free to use any word list you want in the `words.txt` file with each word on one line. Of course, this game cannot work without keeping state information. The player would always have five tries left, could never win without guessing the whole word correctly the first time, and generally, it wouldn't be much fun at all. If you think for a moment about the game's logic, you'll come up with three variables that need to be remembered from request to request:

- The random word the user is to guess.
- The characters the user has already guessed.
- How many tries have failed so far.

Note that all HTML output is handled by a separate template class. This allows the separation of code and layout needed often in professional web applications.

```
require("EasyTemplate.inc.php3");
define("HANG_MAX_TRIES", 6);

$guess = strtolower($guess);

function hang_get_random_line($file)
{
    // Try to open the file
    if(!file_exists($file) || !($fp = fopen($file, "r")))
        die("Could not open file \"$file\" for reading.");

    // Get file size
    $size = filesize($file);

    // Init randomizer, get a random value in the range 0..file size
    srand((double)microtime()*1000000);
    $randval = rand(0, $size);

    // Seek to a random position in the file - possible this is EOF.
    fseek($fp, $randval);

    // Get line - possibly empty (if at EOF)
    $line = trim(fgets($fp, 1024));
    $line = trim(fgets($fp, 1024));

    // Close file
    fclose($fp);

    // If line was empty, try again
    if(empty($line))
    {
        $line = hang_get_random_line($file);
    }

    // Return random line
    return($line);
}

// Initialize session
session_start();
```

```

// Register our variables
session_register("num_of_tries");
session_register("guessed_chars");
session_register("word");

if($REQUEST_METHOD == "POST" && !empty($word) && !empty($guess))
{
    $guessed_chars[] = $guess;
    if(!strstr($word, $guess))
    {
        // Character not found - one try less left
        $num_of_tries++;
        $message = "Wrong guess.";
    }
    else
        $message = "Correct guess!";
}
else
{
    $word = strtolower(hang_get_random_line("words.txt"));
    $message = "Welcome to Hangman!";
    $num_of_tries = 1;
    $guessed_chars = array();
}

$guessed_word = $word;

// Construct the template guess word ("_" in place of not-yet-guessed characters)
for($i=0; $i<strlen($word); $i++)
{
    if(!strstr(implode("", $guessed_chars), $word[$i]))
    {
        $guessed_word = str_replace($word[$i], "_ ", $guessed_word);
    }
}

$button = "Go";

// Check whether the user has won or lost the game
if($guessed_word == $word || $word == $guess)
{
    $message = "Correct guess - you've won!";
    session_destroy();
    $button = "New Game";
    $num_of_tries = 1;
}
elseif($num_of_tries == HANG_MAX_TRIES || strlen($guess) > 1)
{
    $message = "You've lost. The word was \"\$word\".";
    session_destroy();
    $button = "Try Again";
    $num_of_tries = HANG_MAX_TRIES;
}

// Create new template instance
$tpl = new EasyTemplate("template.inc.html");

// Assign template variables
$tpl->assign("ACTION", basename($PHP_SELF));
$tpl->assign("NUM_OF_TRIES", $num_of_tries);
$tpl->assign("BUTTON", $button);

```

```
$tpl->assign("MESSAGE", $message);
$tpl->assign("WORD", $guessed_word);
$tpl->assign("GUESSED_CHARS", @implode("", $guessed_chars));

// Output template
$tpl->easy_print();
```

- (a) [5 points] Explain why the code `$line = trim(fgets($fp, 1024));` is executed twice in the function `hang_get_random_line`.
- (b) [5 points] Explain why the script checks whether it has been invoked by an HTTP POST operation.
- (c) [5 points] Explain how the number of tries could be computed, instead of stored as a session variable.
- (d) [5 points] Explain how the not-yet-guessed letters are kept hidden from the user. Would it be possible for a hacker who can break security on the client, but not on the server, to gain access to these, and hence always get a perfect score?
- (e) [5 points] Explain in English how to extend the code to remember and display a player's best score over multiple games.
- (f) [10 points] Write the PHP code needed to do part (e) and show or explain clearly where it fits into the code above. Details of PHP syntax are not important.