

Final Examination

Thursday June 14, 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 80 points, five questions, and seven pages in total.

Whenever you make an assumption, state it clearly. In questions where you are asked to discuss an issue, it is important that you use the relevant concepts and terminology covered in class and in the textbooks.

You may use a calculator, one PHP book, one MySQL book, the documents linked to the class web page, the published 134A lecture notes, and your own personal notes. No other documents may be used.

(Question 1) [15 points] Your answer to each part of this question should be five to ten sentences.

- (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.
- (b) [5 points] What are the fundamental client-side and server-side differences between a VoiceXML application and a browser-based application? (Assume that the latter uses HTML but not Javascript on the browser.)
- (c) [5 points] How can you maximize the usability of a VoiceXML application? Based on your experience with the third project and elsewhere, explain what you think are the most important VoiceXML usability guidelines.

(Question 2) [25 points] This question is based on exercises due to Peter MacHale. Assume the following database schema:

SALESREP: rep_no rep_name rep_addr

CUSTOMER: cust_no cust_name cust_addr balance rep_no

ORDER: order_no order_date cust_no

ORDER_LINE: order_no part_no quantity_ordered

PART: part_no description units_available category wholesale_price

The type of each column is irrelevant. The meanings of columns should be clear from their names.

For each of the following tasks described in English, give a single correct SQL query. If desired, you may use subselects, even though the current version of MySQL does not provide these.

- (a) [3 points] List the description and part_no of each part with category “automotive.” Include in the result table a column named retail_price, which is wholesale_price with 8% tax added.
- (b) [4 points] For each order_no, find the total number of parts ordered.
- (c) [4 points] Count the number of sales representatives who have handled an order for a part in category “appliance.”
- (d) [4 points] List their name and address for all representatives dealing with the customer(s) with the highest balance.
- (e) [10 points] For each of the select queries you write for parts (a) to (d), explain whether or not one or more indexes could make the query run much faster. If so, explain which index or indexes. Do not suggest indexes not needed for the queries described above.

(Question 3) [10 points] For the first project, to answer a user's query, your script gathered information directly from one or several web sites. For the third project, "the retriever should populate and refresh the database regardless of whether or not the content browser fetches that data."

(a) [6 points] Each design has advantages and disadvantages along each of these dimensions:

- scalability
- response quality for a user
- convenience of implementing alternative presentation media

Explain these advantages and disadvantages.

(b) [4 points] Explain a sensible extension of the third project that would make good use of SOAP, and also of XML for representing data.

(Question 4) [10 points] This question concerns basic PHP programming.

Write a function named `findwords` that returns an array containing all the words of a sentence, which is supplied as a string, with all punctuation removed.

For example, the result of `$a = findwords("hello world, how are you?")` should be `$a[0]=="hello"`, `$a[1]=="world"` and so on.

You must write a function that is simple and efficient, and that uses appropriate built-in PHP functions.

(Question 5) [20 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) [2 points] Explain why the script checks whether it has been invoked by an HTTP POST operation.
- (b) [3 points] Explain how the number of tries could be computed, instead of stored as a session variable.
- (c) [5 points] Explain how the not-yet-guessed letters are kept hidden from the user. Would it be possible for a client-side hacker to gain access to these, and hence always get a perfect score?
- (d) [10 points] Explain how to extend the code to remember and display a player's best score over multiple games. (Write the PHP code needed to do this and show or explain clearly where it fits into the code above. Fine details of PHP syntax are not important.)