

The Zend PHP Certification Practice Test Book

Practice Questions for the
Zend Certified Engineer Exam

John Coggeshall and Marco Tabini





December, 2004

Thank you for your interest in *The Zend PHP Certification Practice Test Book*. This PDF file contains the front matter and a sample chapter from the book.

The book is currently available, both in print and PDF format, exclusively from the php|architect website:

[Click here from the print edition](#)

[Click here for the PDF edition](#)

Effective January 2005, the book will also be available at bookstores throughout North America and Europe, or through online retailers, such as Amazon.com or BarnesAndNoble.com.

If you cannot find it at your local bookstore, please inform them that the book is distributed through Ingram Books and they should be able to order a copy for you.

Thank you!

The php|architect Team
info@phparch.com

THE ZEND PHP CERTIFICATION PRACTICE TEST BOOK

By John Coggeshall and Marco Tabini



The Zend PHP Certification Practice Test Book

Contents Copyright © 2004-2005 John Coggeshall and Marco Tabini – All Right Reserved

Book and cover layout, design and text Copyright © 2004-2005 Marco Tabini & Associates, Inc. – All Rights Reserved

First Edition: January 2005

ISBN 0-9735898-8-4

Produced in Canada

Printed in the United States

No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical reviews or articles.

Disclaimer

Although every effort has been made in the preparation of this book to ensure the accuracy of the information contained therein, this book is provided “as-is” and the publisher, the author(s), their distributors and retailers, as well as all affiliated, related or subsidiary parties take no responsibility for any inaccuracy and any and all damages caused, either directly or indirectly, by the use of such information.

We have endeavoured to properly provide trademark information on all companies and products mentioned in this book by the appropriate use of capitals. However, we cannot guarantee the accuracy of such information.

Marco Tabini & Associates, The MTA logo, php|architect, the php|architect logo, NanoBook and NanoBook logo are trademarks or registered trademarks of Marco Tabini & Associates Inc.

Zend Technologies, the Zend Logo, Zend Certified Engineer, and the Zend Certified Engineer logo are trademarks or registered trademarks of Zend Technologies, Inc. and are used by agreement with the owner.

Bulk Copies

Marco Tabini & Associates, Inc. offers trade discounts on purchases of ten or more copies of this book. For more information, please contact our sales offices at the address or numbers below.

Credits

Written by

John Coggeshall
Marco Tabini

Published by

Marco Tabini & Associates, Inc.
28 Bombay Ave.
Toronto, ON M3H 1B7
Canada

(416) 630-6202
(877) 630-6202 toll free within North America
info@phparch.com / www.phparch.com

Marco Tabini, Publisher

Technical Reviewers

Derick Rethans
Daniel Kushner

Layout and Design

Arbi Arzoumani

Managing Editor

Emanuela Corso

To Daniel Tabini and Diana Katheryn Coggeshall

May we leave you a better world than the one we found.

Table of Contents

FOREWORD	6
ABOUT THE AUTHORS	8
INTRODUCTION	9
<i>Why a Book of Practice Questions?</i>	10
<i>How is this Book Organized?</i>	10
<i>Finding Errata and Discussing Your Concerns</i>	11
<i>Acknowledgements</i>	11
1. PHP PROGRAMMING BASICS	13
Questions.....	14
Answers.....	22
2. OBJECT-ORIENTED PROGRAMMING WITH PHP 4	26
Questions.....	27
Answers.....	36
3. PHP AS A WEB DEVELOPMENT LANGUAGE	39
Questions.....	40
Answers.....	45
4. WORKING WITH ARRAYS	47
Questions.....	48
Answers.....	54
5. STRINGS AND REGULAR EXPRESSIONS	56
Questions.....	57
Answers.....	63
6. MANIPULATING FILES AND THE FILESYSTEM	66
Questions.....	67
Answers.....	73
7. DATE AND TIME MANAGEMENT	76
Questions.....	77
Answers.....	82
8. E-MAIL HANDLING AND MANIPULATION	85
Questions.....	86
Answers.....	91

9. DATABASE PROGRAMMING WITH PHP	94
Questions.....	95
Answers.....	100
10. STREAM AND NETWORK PROGRAMMING.....	102
Questions.....	103
Answers.....	107
11. WRITING SECURE PHP APPLICATIONS	109
Questions.....	110
Answers.....	116
12. DEBUGGING CODE AND MANAGING PERFORMANCE.....	119
Questions.....	120
Answers.....	124

Foreword

There are many advantages to having a PHP certification program. Foremost, it allows employers, especially those of the non-technical kind, to set a certain standard for their PHP hiring decisions; they'll know that people who are certified have passed a set of hurdles in earning their credentials and can clearly demonstrate their knowledge of PHP and its related technologies.

Not only does that mean that a Zend Certified Engineer will automatically match such criteria and have an immediate advantage on the job market, but the certification process also allows for more and more enterprises to adopt PHP. This, in turn, will lead to a much more vibrant job market for PHP developers—making it easier to make a living from what PHP developers like doing most. I have no doubt that we will see an increase in the ongoing PHP proliferation due to the existence of Zend's PHP Certification Exam.

A few weeks ago, I finally found time to take the Zend PHP Certification Exam. Despite having written some of the questions and being part of the exam education advisory board that reviewed the questions a few months ago, I was surprised to realize that I was a tad bit tense—I think not only because exams in general tend to have this effect on me, but also because I

remembered that the questions were very thorough, most probably due to the fact that the exam authors themselves are leaders in the PHP community who wanted to come up with the best possible questions. Without making the exam overly difficult, this ensured that every question was well-thought-out, thoroughly peer-reviewed and carefully constructed; this is bound to make any prospective exam-taker—especially one that was an integral part of such a thorough process—a bit nervous!

I'm happy to say that I passed the exam—but I admit that some questions were quite hard. I think that, overall, the exam is fair but, unlike many other certification tests, much more thorough. A PHP developer with no experience really cannot pass this exam, which I think is great. It really certifies PHP developers who have experience in developing PHP based web applications in the real world.

I believe this book will be of great help in preparing for the certification exam. Both Marco and John were on the Zend PHP Certification Advisory Board and understand the nature of the exam and what its goals are. Both authors also have many years of experience in PHP, which is readily recognizable from the book's contents. This book very nicely covers the different topics on which you will be tested and provides questions that are very similar to the ones you will see on the exam. Having the answers at the end of each chapter will make it easy for you to validate your strengths and weaknesses.

I wish you all the best with the certification progress and hope you will soon join the growing family of Zend Certified Engineers.

Andi Gutmans

Co-founder & VP of Technology, Zend Technologies
Zend Certified Engineer

About the Authors

John Coggeshall is a Technical Consultant for Zend Technologies, where he provides professional services to clients around the world. He got started with PHP in 1997 and is the author of three published books and over 100 articles on PHP technologies with some of the biggest names in the industry such as php|architect, SAMS Publishing, Apress and O'Reilly. John also is an active contributor to the PHP core as the author of the tidy extension, a member of the Zend Education Advisory Board, and frequent speaker at PHP-related conferences worldwide. His web site, <http://www.coggeshall.org/> is an excellent resource for any PHP developer

Marco Tabini is the publisher of php|architect (<http://www.phparch.com>), the premier magazine for PHP professionals. The author and co-author of four books, he was also part of the group of Subject Matter Experts (SMEs) who helped write the Zend Certification Exam. He regularly maintains a blog, which can be found at <http://blogs.phparch.com>, where he discusses the business of open-source software.

Introduction

WRITING AN EXAM IS never an easy task. Socrates is quoted as saying that “an unexamined life is not worth living,” but (although he wasn’t really referring to taking technical tests) we’re sure that most people sitting in an examination room would gladly exchange places with the legendary philosopher and drink his hemlock rather than take a test.

Luckily, writing an exam doesn’t *have* to be such a traumatic experience. Given enough preparation and experience, you should be able to successfully pass it without much in the way of problems. The Zend exam itself is designed with two goals in mind: first, to test your knowledge of PHP and, second, to do so with as much of a practical approach as possible.

The idea of testing only your knowledge of PHP is based on a simple assumption: that your experience as a PHP programmer is not measured by your knowledge of external technologies. As we will reiterate in Chapter 9, you may go all your life developing PHP without ever having to interface to a MySQL database and, therefore, testing your knowledge of MySQL

would be an unfair way to gauge your familiarity with PHP. Besides, MySQL AB (as well as most other relevant vendors of third-party software) already has its own certification program.

As far as the practicality of the questions goes, none of the Subject Matter Experts (SMEs) believed that a good programmer should be a walking PHP reference book. The truth is that PHP provides in excess of 1,500 different functions—and knowing each one of them, together with all its nuances, would be not only practically impossible, but useless as well. The PHP Manual is promptly available online from anywhere in the world; therefore, unless someone is going to lock you in a room with no Internet access, the chances that you won't be able to access it are quite minimal. Still, you can't program if you have to consult the manual every thirty seconds and, therefore, the exam does feature questions that test your knowledge of some basic PHP functionality in a very didactic way.

For the most part, however, the exam tests your ability to understand, interpret and write proper PHP code. Prepare to be asked to analyze plenty of code examples to find out what they do, how they work and whether they have any bugs. Some of the questions may seem a little tricky and unduly complex, but, if you think about it, having to deal with less-than-perfect code (written by someone else, of course!) is not that uncommon for anyone who has ever worked in the real world.

Why a Book of Practice Questions?

It's always best to go into an exam as prepared as possible. Your experience, as well books like the Zend PHP Certification Study Guide, published by SAMS, will be an invaluable tool in ensuring that you will pass, but sitting down and taking the exam itself is unlike anything you're likely to do as part of your daily routine.

This book provides with you a highly structured series of questions designed to mimic (without reproducing) the actual questions that you will find in the exam. It will help you “get in the spirit” of the exam and learn how the questions are phrased and what they expect you to be able to do.

We worked hard at building questions that, while close to the real ones, are usually slightly more difficult to answer correctly. The reasoning behind this is simple: if you can get the hard ones right, the real exam will be a breeze!

How is this Book Organized?

The Zend PHP Certification Practice Test Book is designed to work as a companion to the Official Zend PHP Certification Study Guide (ISBN 0672327090) published by SAMS Publishing. As a result, its chapters closely reflect those of the guide in order to facilitate your learning process as much as possible. You can read a chapter of the guide, then turn to the corresponding chapter in the Practice Questions Book and take a mini-exam centered exclusively on that particular topic. As an alternative, you can use this book as a testing resource together with the PHP Manual. Our table of contents will give you the basic layout of the topics covered by the exam, which you can use to study directly from the manual, as well as many of the other resources available on the Internet. Once you think you're ready to try your hand at some questions, you can use this book again for that purpose.

Each chapter contains fifteen questions, with the exclusion of Chapters 1, 2, 5 and 6, which contain twenty. The reason for this is that these chapters discuss the most fundamental aspects of PHP; therefore, we thought that a few additional questions per chapter might have helped you better gauge your preparedness. You'll find the answers to all the questions, together with an explanation, at the end of each chapter.

While you are, of course, free to use this book any way you like, we'd like to suggest a simple approach that can help you maximize its effectiveness. First of all, you should try your hand at practicing your test when you actually have time to do so—allow at least ninety minutes for answering the questions, and then another thirty to sixty minutes to check your answers.

Start by answering five questions from each fifteen-question chapter, and six from the twenty-question ones. Take care of each chapter in sequence, without stopping to check your answers. This will add up to around sixty-five questions, a very close approximation of the actual exam, which contains seventy. Give yourself around eighty minutes to complete the entire set—again, a good approximation of the ninety minutes allocated in the real exam.

At the end of this process, you can go back and check the answers you gave against the correct ones reported at the end of each chapter. This will give you an opportunity to determine how prepared you are in each different area and to focus your studies on those topics where your results were less than brilliant.

Once you feel ready, you can try again using the same technique. This will make it possible for you to answer a fresh batch of questions every time and test your knowledge anew.

Finding Errata and Discussing Your Concerns

A *lot* of work went into writing, reviewing, editing and then reviewing some more the questions in this book, as well as their answers. Yet, we are but mere mortals and, as such, prone to making mistakes.

If you think you've found something wrong with the contents of the book, come and discuss it on the `php|architect` forums at this URL:

<http://forums.phparch.com/162>

Of course, the same is also true if one of the questions has you stumped and you want to chat with other PHP enthusiasts about the *how's* and the *why's* of the answers we provide. Both of us visit the forums regularly, and we are always happy to help out.

Acknowledgements

Writing a book—no matter how small—is always a monumental task that involves the assistance and expertise of many different people.

We'd like to extend our thanks to Derick Rethans, who has spent considerable time performing a ruthless technical review of each question, pointing out errors and suggesting ways to improve the overall quality of the practice tests. It's thanks to him that so many questions are understandable and technically accurate—and entirely our responsibility if some others are not.

Our thanks also go to Daniel Kushner over at Zend for his unwavering support and invaluable contribution to the Zend Certification Program—without him, there would be no questions to write about.

John Coggeshall
New York City

Marco Tabini
Toronto

1

PHP

Programming

Basics

THE ZEND EXAM IS designed so that you need a reasonable amount of experience in order to pass it. This doesn't mean that you have to be Superman—it simply means that, in order to pass the exam, you've had to have a good amount of exposure to PHP in your daily life.

Therefore, it is essential that you know your “basics” very well. These are the elements of PHP that you will deal with on a constant basis, since they are at the very foundation of the language itself. While not being very prepared on other areas of the exam may only be the result of them not being part of your day-to-day programming routine, failing a considerable number of questions in this chapter should raise a red flag. After all, if you don't know the basics, you'll have trouble understanding more advanced topics as well.

Questions

1. Choose the selection that best matches the following statements:

PHP is a _____ scripting language based on the _____ engine. It is primarily used to develop dynamic _____ content, although it can be used to generate _____ documents (among others) as well.

- A. Dynamic, PHP, Database, HTML
- B. Embedded, Zend, HTML, XML
- C. Perl-based, PHP, Web, Static
- D. Embedded, Zend, Docbook, MySQL
- E. Zend-based, PHP, Image, HTML

2. Which of the following tags is not a valid way to begin and end a PHP code block?

- A. <% %>
- B. <? ?>
- C. <?= ?>
- D. <! !>
- E. <?php ?>

3. Which of the following is not valid PHP code?

- A. \$_10
- B. \${"MyVar" }
- C. &\$something
- D. \$!0_somethi ngs
- E. \$aVaR

4. What is displayed when the following script is executed?

```
<?php
    defi ne(myval ue, "10");

    $myarray[10] = "Dog";
    $myarray[] = "Human";
```

```
$myarray['myvalue'] = "Cat";  
$myarray["Dog"] = "Cat";  
print "The value is: ";  
print $myarray[myvalue]. "\n";
```

?>

- A. The value is: Dog
- B. The value is: Cat
- C. The value is: Human
- D. The value is: 10
- E. Dog

5. What is the difference between `print()` and `echo()`?

- A. `print()` can be used as part of an expression, while `echo()` can't
- B. `echo()` can be used as part of an expression, while `print()` can't
- C. `echo()` can be used in the CLI version of PHP, while `print()` can't
- D. `print()` can be used in the CLI version of PHP, while `echo()` can't
- E. There's no difference: both functions print out some text!

6. What is the output of the following script?

```
<?php  
$a = 10;  
$b = 20;  
$c = 4;  
$d = 8;  
$e = 1.0;  
  
$f = $c + $d * 2;  
$g = $f % 20;  
$h = $b - $a + $c + 2;  
$i = $h << $c;  
$j = $i * $e;  
  
print $j;
```

?>

- A. 128
- B. 42
- C. 242.0
- D. 256
- E. 342

7. Which values should be assigned to the variables \$a, \$b and \$c in order for the following script to display the string Hello, World!?

```
<?php
$string = "Hello, World!";
$a = ?;
$b = ?;
$c = ?;

if($a) {
    if($b && !$c) {
        echo "Goodbye Cruel World!";
    } else if(!$b && !$c) {
        echo "Nothing here";
    }
} else {
    if(!$b) {
        if(!$a && (!$b && $c)) {
            echo "Hello, World!";
        } else {
            echo "Goodbye World!";
        }
    } else {
        echo "Not quite.";
    }
}
?>
```

- A. False, True, False
- B. True, True, False
- C. False, True, True
- D. False, False, True
- E. True, True, True

8. What will the following script output?

```
<?php
$array = '0123456789ABCDEFGH';
$s = '';
for ($i = 1; $i < 50; $i++) {
    $s .= $array[rand(0, strlen($array) - 1)];
}
echo $s;
?>
```

- A. A string of 50 random characters
- B. A string of 49 copies of the same character, because the random number generator has not been initialized
- C. A string of 49 random characters
- D. Nothing, because \$array is not an array
- E. A string of 49 'G' characters

9. Which language construct can best represent the following series of if conditionals?

```
<?php
    if($a == 'a') {
        somefunction();
    } else if ($a == 'b') {
        anotherfunction();
    } else if ($a == 'c') {
        dosomething();
    } else {
        donothing();
    }
?>
```

- A. A switch statement without a default case
- B. A recursive function call
- C. A while statement
- D. It is the only representation of this logic
- E. A switch statement using a default case

10. What is the best way to iterate through the \$myarray array, assuming you want to modify the value of each element as you do?

```
<?php
    $myarray = array ("My String",
                     "Another String",
                     "Hi, Mom!");
?>
```

- A. Using a for loop
- B. Using a foreach loop
- C. Using a while loop
- D. Using a do...while loop
- E. There is no way to accomplish this goal

11. Consider the following segment of code:

```
<?php
    define("STOP_AT", 1024);

    $result = array();

    /* Missing code */
    {
        $result[] = $idx;
    }

    print_r($result);
?>
```

What should go in the marked segment to produce the following array output?

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 4
    [3] => 8
    [4] => 16
    [5] => 32
    [6] => 64
    [7] => 128
    [8] => 256
    [9] => 512
)
```

- A. `foreach($result as $key => $val)`
- B. `while($idx *= 2)`
- C. `for($idx = 1; $idx < STOP_AT; $idx *= 2)`
- D. `for($idx *= 2; STOP_AT >= $idx; $idx = 0)`
- E. `while($idx < STOP_AT) do $idx *= 2`

12. Choose the appropriate function declaration for the user-defined function `is_leap()`. Assume that, if not otherwise defined, the `is_leap` function uses the year 2000 as a default value:

```
<?php
/* Function declaration here */
{
    $is_leap = (!($year % 4) && (($year % 100) ||
        !($year % 400)));
}
```

```

    return $is_leap;
}

var_dump(is_leap(1987));    /* Displays false */
var_dump(is_leap());      /* Displays true */

?>

```

- A. function is_leap(\$year = 2000)
- B. is_leap(\$year default 2000)
- C. function is_leap(\$year default 2000)
- D. function is_leap(\$year)
- E. function is_leap(2000 = \$year)

13. What is the value displayed when the following is executed? Assume that the code was executed using the following URL:

testscript.php?c=25

```

<?php

function process($c, $d = 25)
{
    global $e;
    $retval = $c + $d - $_GET['c'] - $e;
    return $retval;
}

$e = 10;
echo process(5);

?>

```

- A. 25
- B. -5
- C. 10
- D. 5
- E. 0

14. Consider the following script:

```

<?php
function myfunction($a, $b = true)
{
    if($a && !$b) {
        echo "Hello, World!\n";
    }
}

```

```

    }

    $s = array(0 => "my",
              1 => "call",
              2 => '$function',
              3 => ' ',
              4 => "function",
              5 => '$a',
              6 => '$b',
              7 => 'a',

              8 => 'b',
              9 => ' ');

    $a = true;
    $b = false;
    /* Group A */
    $name = $s[?]. $s[?]. $s[?]. $s[?]. $s[?]. $s[?];

    /* Group B */
    $name(${ $s[?] }, ${ $s[?] });

?>

```

Each ? in the above script represents an integer index against the \$s array. In order to display the Hello, World! string when executed, what must the missing integer indexes be?

- A. Group A: 4,3,0,4,9,9 Group B: 7,8
- B. Group A: 1,3,0,4,9,9 Group B: 7,6
- C. Group A: 1,3,2,3,0,4 Group B: 5,8
- D. Group A: 0,4,9,9,9,9 Group B: 7,8
- E. Group A: 4,3,0,4,9,9 Group B: 7,8

15. Run-time inclusion of a PHP script is performed using the _____ construct, while compile-time inclusion of PHP scripts is performed using the _____ construct.

- A. include_once, include
- B. require, include
- C. require_once, include
- D. include, require
- E. All of the above are correct

16. Under what circumstance is it impossible to assign a default value to a parameter while declaring a function?

- A. When the parameter is Boolean
- B. When the function is being declared as a member of a class
- C. When the parameter is being declared as passed by reference
- D. When the function contains only one parameter
- E. Never

17. The ____ operator returns True if either of its operands can be evaluated as True, but not both.

Your Answer: _____

18. How does the identity operator === compare two values?

- A. It converts them to a common compatible data type and then compares the resulting values
- B. It returns True only if they are both of the same type and value
- C. If the two values are strings, it performs a lexical comparison
- D. It bases its comparison on the C strcmp function exclusively
- E. It converts both values to strings and compares them

19. Which of the following expressions multiply the value of the integer variable \$a by 4?
(Choose 2)

- A. `$a *= pow (2, 2);`
- B. `$a >>= 2;`
- C. `$a <<= 2;`
- D. `$a += $a + $a;`
- E. None of the above

20. How can a script come to a clean termination?

- A. When `exit()` is called
- B. When the execution reaches the end of the current file
- C. When PHP crashes
- D. When Apache terminates because of a system problem

Answers

1. Looking at the answers, the only one that makes sense for every blank is B. PHP is a scripting language based on the Zend Engine that is usually embedded in HTML code. As such, it is primarily used to develop HTML documents, although it can be used just as nicely to develop other types of documents, such as XML.
2. While tags such as `<% %>` and `<?= ?>` are often forgotten in PHP programming, they are valid ways to delimit a PHP code block. The `<!>` and `<!>` tags, however, are not valid and, therefore, the correct answer is D. Keep in mind, in any case, that some of these tags are not always available, depending on how the `php.ini` file on which the PHP interpreter runs is configured.
3. PHP variables always start with a dollar sign and are a sequence of characters and numbers within the Latin alphabet, plus the underscore character. `MyVar` is a valid variable name that simply uses a slightly less common naming convention, while `&$something` is a reference to the `$something` variable. Variables, however cannot start with numbers, making `$10_somethings` invalid and Answer D correct.
4. The important thing to note here is that the `$myarray` array's key value is being referenced without quotes around it. Because of this, the key being accessed is not the `myvalue` string but the value represented by the `myvalue` constant. Hence, it is equivalent to accessing `$myarray[10]`, which is Dog, and Answer A is correct.
5. Even though `print()` and `echo()` are essentially interchangeable most of the time, there is a substantial difference between them. While `print()` behaves like a function with its own return value (although it is a language construct), `echo()` is actually a language construct that has no return value and cannot, therefore, be used in an expression. Thus, Answer A is correct.
6. Other than the simple math, the `%` operator is a modulus, which returns whatever the remainder would be if its two operands were divided. The `<<` operator is a left-shift operator, which effectively multiplies an integer number by powers of two. Finally, the ultimate answer is multiplied by a floating point and, therefore, its type changes accordingly. However, the result is still printed out without any fractional part, since the latter is nil. The final output is 256 (Answer D).
7. Following the logic of the conditions, the only way to get to the `Hello, World!` string is in the `else` condition of the first `if` statement. Thus, `$a` must be `False`. Likewise, `$b` must be `False`. The final conditional relies on both previous conditions (`$a` and `$b`) being `False`, but insists that `$c` be `True` (Answer D).
8. The correct answer is C. As of PHP 4.2.0, there is no need to initialize the random number generator using `srand()` unless a specific sequence of pseudorandom numbers is sought.

Besides, even if the random number generator had not been seeded, the script would have still outputted 49 pseudo-random characters—the same ones every time. The `$array` variable, though a string, can be accessed as an array, in which case the individual characters corresponding to the numeric index used will be returned. Finally, the `for` loop starts from 1 and continues until `$i` is less than 50—for a total of 49 times.

9. A series of `if...else if` code blocks checking for a single condition as above is a perfect place to use a `switch` statement:

```
<?php
    switch($a) {
        case 'a':
            somefunction();
            break;
        case 'b':
            anotherfunction();
            break;
        case 'c':
            dosomething();
            break;
        default:
            donothing();
    }
?>
```

Because there is a catch-all else condition, a default case must also be provided for that situation. Answer E is correct.

10. Normally, the `foreach` statement is the most appropriate construct for iterating through an array. However, because we are being asked to modify each element in the array, this option is not available, since `foreach` works on a copy of the array and would therefore result in added overhead. Although a `while` loop or a `do...while` loop might work, because the array is sequentially indexed a `for` statement is best suited for the task, making Answer A correct:

```
<?php
    $myarray = array ("My String", "Another String", "Hi, Mom!");
    for($i = 0; $i < count($myarray); $i++)
    {
        $myarray[$i] .= " ($i)";
    }
?>
```

11. As it is only possible to add a single line of code to the segment provided, the only statement that makes sense is a `for` loop, making the choice either C or D. In order to select the `for`

loop that actually produces the correct result, we must first of all revisit its structural elements. In PHP, for loops are declared as follows:

```
for (<init statement>; <continue until statement>;
    <iteration statement>)
```

where the <init statement> is executed prior to entering the loop. The for loop then begins executing the code within its code block until the <continue until> statement evaluates to False. Every time an iteration of the loop is completed, the <iteration statement> is executed. Applying this to our code segment, the correct for statement is:

```
for ($idx = 1; $idx < STOP_AT; $idx *= 2)
```

or answer C.

12. Of the five options, only two are valid PHP function declarations (A and D). Of these two declarations, only one will provide a default parameter if none is passed—Answer A.
13. This question is designed to test your knowledge of how PHP scopes variables when dealing with functions. Specifically, you must understand how the `global` statement works to bring global variables into the local scope, and the scope-less nature of superglobal arrays such as `$_GET`, `$_POST`, `$_COOKIE`, `$_REQUEST` and others. In this case, the math works out to $5 + 25 - 25 - 10$, which is -5 , or answer B.
14. Functions can be called dynamically by appending parentheses (as well as any parameter needed) to a variable containing the name of the function to call. Thus, for Group A the appropriate index combination is 0, 4, 9, 9, 9, 9, which evaluates to the string `myfunction`. The parameters, on the other hand, are evaluated as variables dynamically using the `${}` construct. This means the appropriate indexes for group B are 7 and 8, which evaluate to `['a']` and `['b']`—meaning the variables `$a` and `$b` respectively. Therefore, the correct answer is D.
15. In recent versions of PHP, the only difference between `require()` (or `require_once()`) and `include()` (or `include_once()`) is in the fact that, while the former will only throw a warning and allow the script to continue its execution if the include file is not found, the latter will throw an error and halt the script. Therefore, Answer E is correct.
16. When a parameter is declared as being passed by reference you cannot specify a default value for it, since the interpreter will expect a variable that can be modified from within the function itself. Therefore, Answer C is correct.
17. The right answer here is the exclusive-or (`xor`) operator.
18. The identity operator works by first comparing the type of both its operands, and then their values. If either differ, it returns False—therefore, Answer B is correct.

- 19.** The correct answers are A and C. In Answer A, the `pow` function is used to calculate 2^2 , which corresponds to 4. In Answer C, the left bitwise shift operator is used to shift the value of `$a` by two bits to the left, which corresponds to a multiplication by 4.
- 20.** The only answer that really fits the bill is A. A script doesn't necessarily terminate when it reaches the end of any file other than the main one—so the “current” file could be externally included and not cause the script to terminate at its end. As far as PHP and Apache crashes, they can hardly be considered “clean” ways to terminate a script.

