ITプロ／ITエンジニアのための

徹底攻略

試験番号
PJ0-200

PHP技術者認定機構
公式
問題集

PHP5

［PJ0-200］対応

技術者認定［上級］試験

問題集

鈴木憲治／堀 邦彰／水野史土／山田裕進 著　PHP 技術者認定機構 監修
株式会社ソキウス・ジャパン 編

インプレスジャパン

本書は、PHP5技術者認定上級試験の受験対策用の教材です。著者、株式会社インプレスジャパンは、本書の使用による同試験への合格を一切保証しません。

本書の内容については正確な記述につとめましたが、著者、株式会社インプレスジャパンは本書の内容に基づく試験の結果にも一切責任を負いません。

本文中の製品名およびサービス名は、一般に各開発メーカーおよびサービス提供元の商標または登録商標です。なお、本文中にはTMおよび®は明記していません。

インプレスジャパンの書籍ホームページ

書籍の新刊や正誤表など最新情報を随時更新しております。

http://www.impressjapan.jp/

Web徹底攻略

試験や資格の最新情報や模擬試験などが体験できる資格関連書の専用サイトです。

http://shikaku.impress.co.jp/

Copyright © 2013 Socius Japan, Inc. All rights reserved.
本書の内容はすべて、著作権法によって保護されています。著者および発行者の許可を得ず、転載、複写、複製等の利用はできません。

はじめに

PHP（PHP：Hypertext Preprocessor）は、広く利用されているオープンソースの汎用スクリプト言語の1つです。HTMLに埋め込むことができるためウェブの開発に適しており、世界中で多くのウェブサイト構築に用いられています。ウェブアプリケーションの開発に携わるエンジニアのみならず、ウェブサイトに動的な仕組みを導入したいウェブデザイナーまで、幅広いユーザによって利用されています。

PHP技術者認定試験とは、特定非営利活動任意団体PHP技術者認定機構により運営・認定されている、ベンダーに依存しない認定試験制度です。PHPの専門技術取得能力を正当に評価する指標を提供することにより、「認定者の雇用機会」や「認定者が所属する会社のビジネスチャンス」の拡大を図ることを目的としています。試験合格を目標に学習することで、PHPを扱う上で必要な知識を体系立てて習得することができ、資格を取得すればPHPプログラミングに必要となる幅広い知識を持っていることを客観的に証明できます。

本書はPHP技術者認定試験のうち、「PHP5技術者認定上級試験」の試験対策に即した問題集です。PHPを用いた開発経験のある中級以上の方を対象にしています。この試験に合格するには、PHPの言語仕様だけではなく、実用的なウェブサイトを構築するために必要となる技術やライブラリについての幅広い知識が必要です。

執筆にあたっては、PHP上級試験で出題される範囲を網羅するように配慮するとともに、PHPの開発に携わるエンジニアが知っておくべき内容についても、ふんだんなサンプルコードとともに幅広く解説に取り入れるようにしています。特にセキュリティについては、昨今の関心の高さや、試験の出題範囲全体にわたり関連していることから、第10章「セキュリティ」で詳細に取り上げるとともに、その他の章でも取り上げています。また、解説は正解を示すだけでなく、設問に関連する知識を習得できるように詳細に記述しています。

本書の執筆にあたっては、株式会社ソキウス・ジャパンならびに関係者の方々に大変お世話になりました。ここに感謝いたします。

最後に、本書が試験合格への最短経路として、また、PHP技術者のスキルアップのためにお役に立つことを心より願っております。

2013年7月 著者一同

PHP5技術者認定試験について

プログラミングの技術力の判断基準によく用いられるものに、業務年数と携わったプロジェクトの内容があります。しかしこれらは基準があいまいで、客観的な指標とすることは困難です。

たとえばPHPの場合、Web開発の業務年数やプロジェクト内容だけでは、どの程度PHPを使いこなせるかを正しく評価することはできません。業務とは別に独学で技術を習得した人や学生にとっては、これは切実な問題です。

そのような問題を解決するために、PHP技術者認定機構は、2011年、PHP技術の習得度を正当に評価できる技術者認定試験を開始しました。この試験は、日本市場においてPHP技術者の育成を図ると同時に、技術者の雇用機会や、技術者を擁する企業のビジネスチャンスを拡大することをも目的としています。

PHP技術者認定試験は、初級、上級、ウィザードの3つのレベルから構成されており、受験者の技量に合わせて、初級もしくは上級を選択して受験することができます。

PHP5技術者認定初級試験

初級試験では主にPHPプログラミングの基礎知識が問われます。

変数や式、関数などの構文から、HTTPに関する知識、データベースの扱い方、配列の定形処理や、正規表現などを用いたテキスト処理、セキュリティなどについて、PHPに限らずWeb開発において必須となる知識全般が対象になります（試験範囲は7ページを参照）。

これからWeb系の言語を初めて学習する人やデザイナーにとっては、PHP技術者認定初級試験の試験範囲は、何をどこまで学べばよいのかを示してくれるよい指針になるはずです。

また、PHPは記述が容易なので、初心者は思わぬところでセキュリティホールを作ってしまうことがあります。試験範囲を漏れなく学習することで、Web開発を行う上では必須である基本的なセキュリティの知識も身につけることができます。

PHP5技術者認定上級試験

本書が扱う上級試験は、言語仕様や実用的なテクニック・知識の有無を問われる試験です。開発初心者を脱却したい人や、より深くPHPを理解したい人にはよい目標になるでしょう。上級試験の特徴は、70%以上の得点で「上級試験合格」と判定されるだけでなく、それに満たなくても得点に応じてレベルが認定される点です。得点の認定レベルは、以下のようになっています。

●スコア認定

・90%以上（Honor Expert認定）※1
・70%以上～90%未満（Expert認定）※1
　PHPの言語仕様から実用的で高度なプログラミングテクニックをもつ上級者
・50%以上～70%未満（Senior）※2
　オンラインマニュアルなどを参考にプログラミングができるレベル
・30%以上～50%未満（Junior）※2
　オンラインマニュアルなどを参考にプログラミングができるが、部分的に基礎知識の修得が必要なレベル
・30%未満（Sophomore）※2
　基礎知識の修得を必要とするレベル

※1：合格認定証明書が送付されます。
※2：スコア証明書が送付されます。

選択式試験で、得点をスコアとして履歴書に書くことができるので、初級合格直後から受験をしながら、レベルアップを図ることも可能です。

●学習の参考になるリソース

・PHPオンラインマニュアル
　http://php.net/manual/ja/index.php
・@IT模擬問題
　http://jibun.atmarkit.co.jp/scenter/ittrain/124_today_q.html

そのほか、PHP技術者認定機構の下記ページに、参考になるURLが記載されています。

http://www.phpexam.jp/summary/expert/

PHP技術者認定ウィザード

ウィザードとは魔術師の意味で、PHP技術者認定ウィザードは、まさに魔術師のように自由にPHPを操ることができる最上位の技術者の証となる認定資格です。PHP5技術者認定上級試験に合格してから2年以内の受験者が対象で、セキュリティ・ウィザード、パフォーマンス・ウィザード、フレームワーク・ウィザード、インターナル・ウィザードの4つのカテゴリがあります。

ウィザード試験の特徴は、論文もしくはコードの審査によって認定される点で、審査は、PHP技術者認定取得者、PHP技術者認定機構の顧問および役員、PHP技術者認定機構認定スクールによって公開で行われます。詳細は、PHP技術者認定機構の公式Webサイトを参照してください。

PHP5技術者認定上級試験の概要と出題範囲

本書が扱うPHP5技術者認定上級試験の概要と出題範囲は以下のとおりです。

● PHP5 技術者認定上級試験（PJ0-200）

- 問題数　　：60問
- 試験時間　：2時間
- 合格ライン：70％以上正解
- 出題形式　：CBTによる選択式（複数または単一選択）
- 受験料　　：15,000円（税抜）

●出題範囲

『プログラミングPHP 第2版』（2007年、オライリー・ジャパン発行）を主教材とし、一般的な知識やPHPオンラインマニュアルなどからも出題します。出題範囲、出題比率、本書で扱う章は次ページの表のとおりです。

【出題範囲】

章番号	本書の章番号	内容	出題比率
1	—	PHPについて	0
2	—	PHP言語の基本	0
3	1	関数	0.02
4	2	文字列	0.02
5	3	配列	0.04
6	4	オブジェクト	0.1
7	5	ウェブに関するテクニック	0.12
8	6	データベース	0.08
9	7	グラフィック	0.04
10	8	PDF	0.04
11	9	XML（&XML系で追加された関数）	0.06
12	10	セキュリティ	0.12
13	11	アプリケーションに関するテクニック	0.08
14	12	PHPの拡張	0.04
15	13	WindowsでのPHP	0.02
追加1	14	SPL（Standard PHP Library）	0.04
追加2	15	名前空間、クロージャー、リフレクション、Late Static Binding	0.04
追加3	16	JSON	0.02
追加4	17	PDOとネイティブモジュールの違い、mysqlndドライバについて	0.04
追加5	18	正規表現（pcre, posix, mbstring）	0.02
追加6	19	APD／Xdebug、memcache	0.04
追加7	20	フィルター	0.02

出題範囲の1章から15章までの詳細は『プログラミングPHP 第2版』の目次をご覧ください。追加1〜7はオンラインマニュアルなどから出題されます。

主な対象バージョンはPHP5.3です（2013年8月現在）。

試験範囲および出題率は予告なく変更される場合があります。受験前には必ずPHP技術者認定機構の公式Webサイトを確認してください。

受験申し込み方法

PHP技術者認定上級試験は、プロメトリック株式会社のテストセンターで受験することができます。

●プロメトリック株式会社

プロメトリックのテストセンターで随時受験可能です。

URL：http://it.prometric-jp.com/testlist/php/index.html

Tel：0120-347-737

※Webで申し込む場合は、プロメトリックIDを取得する必要があります。

PHP5技術者認定試験の問い合わせ先

● PHP 技術者認定機構

・公式Webサイト　http://www.phpexam.jp/

・公式Twitter　@phpexam

●問い合わせ

以下のサイトの問い合わせフォームからお問い合わせください。

http://www.phpexam.jp/contact/

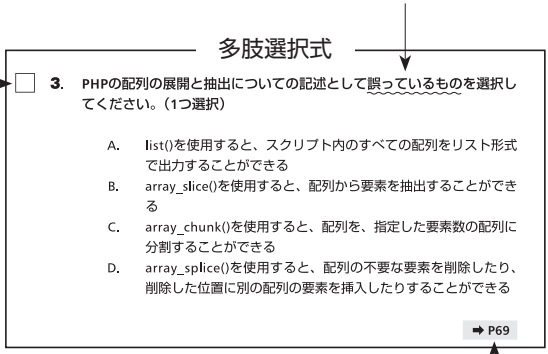
本書の活用方法

本書は、カテゴリ別に分類された、問題と解答で構成されています。

問題

本書の問題は、PHP5 技術者認定上級試験合格に必要な知識を効率的に学習することを目的に作成したものです。解答していくだけで、合格レベルの実力が身につきます。

「誤っているものを選択」する問題も織り交ぜています。
問題文をよく読んで、主旨に合った解答を選択します。



チェックボックス

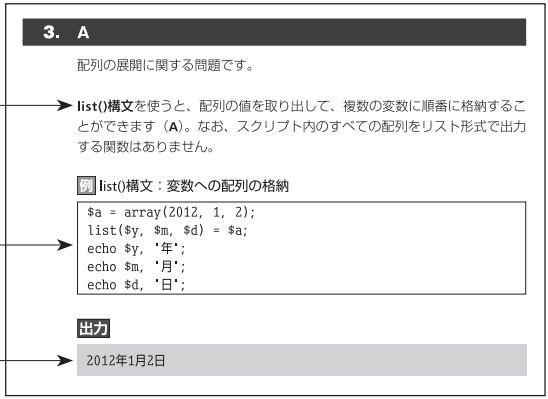
確実に理解している問題のチェックボックスを塗りつぶしながら問題を解き進めると、2回目からは不確かな問題だけを効率的に解くことができます。すべてのチェックボックスが塗りつぶされれば合格は目前です。

解答ページ

問題の右下に、解答ページが表示されています。ランダムに問題を解くときも、解答ページを探すのに手間取ることがありません。

解答

解答には、問題の正解および不正解の理由だけでなく、用語や重要事項などが詳しく解説されています。



解説（用語）

重要な用語やキーワードと、正解の選択肢は「選択肢**A**」のように太字で示しています。

解説（サンプル）

ソースコードの例と実行結果を掲載しています。ソースコードの動作検証はPHP 5.3.24で行っています。

目次

はじめに …… 3
PHP5技術者認定試験について …… 4
PHP5技術者認定上級試験の概要と出題範囲 …… 6
受験申し込み方法 …… 8
PHP5技術者認定試験の問い合わせ先 …… 8
本書の活用方法 …… 9

第1章 関数

問題 …… 14
解答 …… 17

第2章 文字列

問題 …… 32
解答 …… 35

第3章 配列

問題 …… 58
解答 …… 64

第4章 オブジェクト

問題 …… 94
解答 …… 102

第5章 ウェブに関するテクニック

問題 …… 146
解答 …… 157

第6章 データベース

問題 …… 176
解答 …… 184

第7章 グラフィックス
問題 …… 232
解答 …… 237

第8章 PDF
問題 …… 276
解答 …… 279

第9章 XML
問題 …… 290
解答 …… 300

第10章 セキュリティ
問題 …… 320
解答 …… 328

第11章 アプリケーションに関するテクニック
問題 …… 350
解答 …… 361

第12章 PHPの拡張
問題 …… 380
解答 …… 383

第13章 WindowsでのPHP
問題 …… 396
解答 …… 398

第14章 追加1（SPL、日付クラス、PEAR）
問題 …… 402
解答 …… 407

第15章 追加2(名前空間、クロージャ、リフレクション、遅延静的束縛)
問題 …… 426
解答 …… 438

第16章 追加3(JSON)
問題 …… 454
解答 …… 457

第17章 追加4(PDO、mysqlnd)
問題 …… 470
解答 …… 475

第18章 追加5(正規表現)
問題 …… 482
解答 …… 486

第19章 追加6(APD／Xdebug、memcache)
問題 …… 524
解答 …… 528

第20章 追加7(フィルタ)
問題 …… 548
解答 …… 555

第21章 総仕上げ問題
問題 …… 568
解答 …… 606

索引 …… 641

第1章

関数

- 関数の呼び出し
- 関数の定義
- 変数のスコープ
- 関数のパラメータ
- 返り値
- 可変関数
- 匿名関数

☐ **1.** **PHPの関数についての記述として正しいものを選択してください。（3つ選択）**

A. 関数に対する入力を受ける変数をパラメータ、関数からの出力を返り値と呼ぶ
B. 関数には、組み込み関数とユーザ定義関数の2種類がある
C. 関数は必ず返り値を持つ
D. 関数を定義するコードの前には、その関数を呼び出すコードを記述できない
E. 関数には、副作用を持つものがある

➡ P17

☐ **2.** **PHPの関数についての記述として誤っているものを選択してください。（1つ選択）**

A. 数字で始まる関数名の関数は定義できない
B. 関数名の大文字・小文字は区別されるため、たとえば、max()という関数をMAX()という記述で呼び出すことはできない
C. 関数の中には、複数のreturn文を書くことができる
D. 関数の中に、別の関数を定義するコードを記述することができる
E. PHPのコード中に、PHP終了タグ（?>）とPHP開始タグ（<?php）がこの順で現れた場合、その2つのタグに囲まれたテキストはそのまま出力される

➡ P18

☐ **3.** **関数のスコープについての記述として誤っているものを選択してください。（2つ選択）**

A. 関数の外側で宣言された変数は、グローバルスコープの変数となる
B. 関数の内側で宣言された変数は、関数レベルのスコープの変数となる
C. 下記のコードは「1」を出力する

```
function func() {
  echo $x;
}
$x = 1;
func();
```

D. 下記のコードは「1」を出力する

```
function func() {
  $x = 1;
}
func();
echo $x;
```

E. 関数内の変数は、関数の呼び出しが完了すると同時に内容が消えてしまう。staticキーワードを付けることで、関数の呼び出しが完了しても、関数内の変数の値を保持させることができる

➡ P19

☐ **4.** 関数のパラメータについての記述として誤っているものを選択してください。（1つ選択）

A. 関数のパラメータには、値渡しのパラメータと、参照渡しのパラメータの2種類がある

B. パラメータを参照渡しにするには、関数定義のパラメータの前に「*」を記述する

C. パラメータが参照渡しの場合、関数が、引数として与えた変数の内容を変更する可能性がある

D. パラメータが参照渡しの場合、対応する引数に変数を与えないとエラーになる

➡ P21

☐ **5.** パラメータのデフォルト値についての記述として正しいものを選択してください。（1つ選択）

A. パラメータにはデフォルト値を指定することができる。デフォルト値を持つパラメータについては、その関数の呼び出し時に、引数の指定を省略することができる

B. パラメータを参照渡しにした際は、そのパラメータに対するデフォルト値は指定できない

C. 次のコードは「123」を出力する

```
function func($x, $y = 2, $z = 3) {
  echo $x, $y, $z;
}
func(1,,3);
```

➡ P22

☐ **6.** 関数のパラメータについての記述として誤っているものを選択してください。（1つ選択）

A. 関数に定義されたパラメータの数よりも引数の数が少ない場合、エラーが発生し、スクリプトの実行が停止される
B. 関数が不特定の数のパラメータを必要とする場合は、可変パラメータを用いる
C. 可変パラメータを用いる際、関数の呼び出し時に与えられた引数の数を調べるには、func_num_args()を使用する
D. 可変パラメータを用いる際、関数の呼び出し時に与えられた引数の値を得るには、func_get_arg()やfunc_get_args()を使用する

➡ P24

☐ **7.** 関数の返り値についての記述として誤っているものを選択してください。（1つ選択）

A. 関数からの返り値は、returnキーワードを使用して指定する
B. 関数から複数の値を返したい場合は、returnの後ろに複数の値をカンマ区切りで書く
C. デフォルトでは、関数からの返り値は値のコピーとなる
D. 関数が参照を返すようにするには、関数を定義する際、関数名の前に「&」を付ける

➡ P25

☐ **8.** 可変関数および匿名関数についての記述として誤っているものを選択してください。（1つ選択）

A. 可変関数の仕組みを使用すると、関数の名前を変数に代入することで、呼び出す関数を指定できる
B. 可変関数を使用する際、指定された名前の関数が存在しない場合はエラーとなる
C. 可変関数の仕組みを用いてecho()を呼び出すことができる
D. 匿名関数は、その場限りで使用されるような一時的な関数が必要な際に利用される
E. 匿名関数は、create_function()を用いて作成することができる

➡ P27

第 1 章　関数
解　答

1. A、B、E

関数に関する基本的な知識の確認問題です。

関数とは、ひとまとまりのコードに名前を付けたものです。いったん関数を定義すると、その名前を用いて処理を組み立てることができるので、わかりやすく、再利用性の高いコードを記述することができます。

関数に対する入力を受ける変数を**パラメータ**（**仮引数**）、関数からの出力を**返り値**と呼びます（**A**）。次は、2つのパラメータと返り値を持つ関数を定義する例です。

例 2つのパラメータと返り値を持つ関数の定義

```
function img($file, $alt) {
  $s = '<img';
  $s .= ' src="' . $file . '"';
  $s .= ' alt="' . $alt . '"';
  $s .= ' />';
  return $s;
}
```

PHPの関数には、上記のようにユーザが必要に応じて定義する**ユーザ定義関数**と、PHPインタプリタに事前に組み込まれた**組み込み関数**の2種類があり、どちらも同じように呼び出しを行うことができます（**B**）。

関数を呼び出す際、パラメータに対して**値**（**実引数**）を与えます。次は、上記のimg()関数の呼び出しを行う例です。'summer.png'と'summer'が実引数です。

例 img()関数の呼び出し

```
echo img('summer.png', 'summer');
```

出力

```
<img src="summer.png" alt="summer" />
```

関数によっては、パラメータや返り値がないものもあります（C）。次のユーザ定義関数は、パラメータも返り値も持たない例です。

例 パラメータも返り値も持たない関数

```
function echo_top_link() {
  echo '<a href="index.html">top page</a>';
}
```

次のコードのように、関数の呼び出しがその関数の定義よりも前に記述されていてもかまいません（D）。

例 定義より前に呼び出しが記述された関数

```
func();

function func() {
  echo 'OK';
}
```

出力

```
OK
```

関数の中には、**副作用**（引数から返り値を計算する以外の動作をすること）を持つものもあります（**E**）。前述のimg()は、副作用を持たない関数の例です。一方、echo_top_link()、func()は、「テキストを出力する」という副作用を持つ関数の例です。組み込み関数のunlink()は、「ファイルシステムからファイルを削除する」という副作用を持ちます。
PHPに用意されている組み込み関数とそのパラメータ、返り値、処理内容、副作用については、PHPマニュアル（http://php.net/manual/ja/）で確認することができます。

2. B

関数に関する基本的な知識の確認問題です。

関数の名前は、大文字・小文字が区別されません（**B**）。たとえば、引数の最大値を返すmax()という組み込み関数の呼び出しを行う際、すべて小文字でmax()と書いても、すべて大文字でMAX()と書いても、大文字・小文字を混ぜてMax()と書いてもかまいません。一方、変数名は大文字・小文字が区別されるので、注意してください。

関数名の最初の文字は、文字（正確には、正規表現で表すと[a-zA-Z\x7f-\xff]）、または、アンダースコア（_）にする必要があります（A）。2文字目以降には数字も利用できます。

関数の中に複数のreturn文があってもかまいません（C）。関数に複数のreturn文があるときは、その関数が呼び出されたときに最初に到達したreturn文のところで呼び出しが終わります。

関数の中に、別の関数を定義するコードを記述することができます（D）。これを**入れ子**の定義と呼びます。入れ子にされた関数を呼び出すには、先に、それを含む関数を呼び出さなければなりません。次の例では、x()を呼び出した時点でy()が定義され、y()の呼び出しができるようになります。y()のスコープがx()内に限定されることはありません。

例 入れ子の関数

```
function x() {
  function y() {
    echo 'y';
  }
}
```

PHPのコード中に、**PHP終了タグ**（?>）と**PHP開始タグ**（<?php）がこの順で現れた場合、その2つのタグに囲まれたテキストはそのまま出力されます（E）。PHPコード内で大量のHTMLコードを出力したい場合は、echoを繰り返し記述する代わりにこの記法を使用することができます。次の例では、定型的なHTMLコードを出力する関数を定義しています。

例 HTMLコードを出力する関数の定義

```
<?php function print_footer() { ?>
  <span class="copyright">
    Copyright (C) 2013 Sample-corp. All rights reserved.
  </span>
<?php } ?>
```

3. C、D

変数のスコープと、スタティック変数に関する問題です。

コード上における変数の有効範囲のことを**スコープ**といいます。スコープには、**グローバルスコープ**と**関数レベルのスコープ**の2種類があります。変数の宣言位置により、その変数のスコープが決定されます（A、B）。

【変数の宣言位置とスコープ】

宣言位置	種類	有効範囲
関数の外側	グローバルスコープ	すべての位置
関数の内側	関数レベルのスコープ	その関数内

デフォルトでは、関数内からグローバルスコープの変数にアクセスすることはできません（**C**）。関数内からグローバルスコープの変数にアクセスしたい場合は、関数内で**global**キーワードを使用して、その変数を指定しておく必要があります。
選択肢Cのコードは、次のようにglobalキーワードを追加すれば、グローバルスコープの変数**$x**にアクセスすることができます。

例 関数内からグローバルスコープの変数へのアクセス

```
function func() {
  global $x;
  echo $x;
}
$x = 1;
func();
```

出力

```
1
```

関数レベルのスコープの変数には、関数の外部からアクセスすることはできません（**D**）。ただし、return文を使用して変数の値を返すことは可能です。たとえば、選択肢Dのコードで1が表示されるようにするには、次のように書き換えます。

例 return文を使用して値を返す

```
function func() {
  $x = 1;
  return $x;
}
$x = func();
echo $x;
```

出力

```
1
```

関数内の変数は、関数の呼び出しが完了するときに消えてしまいます。**static**キーワードを付けることで、関数の呼び出しが完了しても変数の値を保持させることができます（E）。

例 staticキーワードによる変数の値の保持

```
function func() {
  static $x = 0;
  $x++;
  echo $x, "\n";
}
func();
func();
func();
```

出力

```
1
2
3
```

staticキーワードが付いた変数の代入は、その関数の最初の呼び出しのときだけ実行されます。

staticキーワードが付いた変数は、リテラルあるいは定数を用いて初期化することができます。式で初期化することはできません。

例 式による初期化（エラー）

```
function func($x) {
  static $y = $x; // 式で初期化することはできません
}
```

4. B

値渡しと参照渡しのパラメータ（引数）に関する問題です。

パラメータには、**値渡しのパラメータ**と**参照渡しのパラメータ**の2種類があります（A）。次のコードを見てください。参照渡しを使う場合は、パラメータの前にアスタリスク（*）ではなくアンパサンド（&）を記述します（**B**）。

※次ページに続く

例 値渡しのパラメータ

```
function birthday($age) {
  $age++;
  echo $age . '歳の誕生日おめでとう' ;
}
$x = 20;
birthday($x);
```

例 参照渡しのパラメータ

```
function birthday(&$age) {
  $age++;
  echo $age . '歳の誕生日おめでとう';
}
$x = 20;
birthday($x);
```

値渡しでは、引数の値（20）のコピーがパラメータ（$age）に渡されるため、引数として与えた変数（$x）の内容が関数内で変更されるおそれはありません。上記の値渡しのコードでは、birthday()の呼び出しの完了後、$xの値は20のままです。

一方、参照渡しでは、パラメータ（$age）が、引数として与えた変数（$x）そのものを指し示すようになります。したがって、関数内でのパラメータに対する変更は、対応する引数へ及びます。そのため、引数として与えた変数の値が変更される可能性が生じます（C）。上記の参照渡しのコードでは、birthday()の呼び出しの完了後、$xの値は21となります。
パラメータが参照渡しのとき、引数に変数を使用しないとエラーになります（D）。たとえば、参照渡しのパラメータを持つ関数 `birthday(&$age) { ... }` を `birthday(20);` と記述して呼び出すことはできません。
参照渡しを使用すると、値渡しの場合と違い、値のコピーを作成しません。

5. A

パラメータのデフォルト値に関する問題です。

「**パラメータ = 値**」という構文で、パラメータに**デフォルト値**を指定することができます。次のコードでは、$yと$zにデフォルト値を指定しています。

例 デフォルト値の指定

```
function func($x, $y = 22, $z = 33) {
  echo '$x=', $x;
  echo ', $y=', $y;
  echo ', $z=', $z;
}
```

デフォルト値が指定されているパラメータは、呼び出し時に、対応する引数の記述を省略することができます（**A**）。省略された引数には、デフォルト値で指定された値が補われます。次は、上記のfunc()を、引数の数を変えながら呼び出す例です。

【func()の例】

コード	出力
func(1, 2, 3);	$x = 1, $y = 2, $z = 3
func(1, 2);	$x = 1, $y = 2, $z = 33
func(1);	$x = 1, $y = 22, $z = 33

関数の定義において、あるパラメータにデフォルト値を指定した場合は、それ以降の（右側の）パラメータにもすべてデフォルト値を指定します。

パラメータを参照渡しにした場合でも、デフォルト値の指定は可能です（B）。次のコードは、参照渡しのパラメータにデフォルト値を指定する例です。

例 参照渡しのパラメータへのデフォルト値の指定

```
function func(&$x = 1) {
  $x++;
  echo $x;
}

func();
```

出力

```
2
```

関数の呼び出しの際、デフォルト値を持つパラメータに対する引数の指定を省略した場合は、それ以降の引数の指定もすべて省略しなければなりません。次の例では、第2引数を省略していますが第3引数を指定しているので、エラーとなります（C）。

※次ページに続く

例 引数の指定方法の誤り

```
function func($x, $y = 2, $z = 3) {
  echo $x, $y, $z;
}
func(1,,3);    // エラー（Parse Error）が発生します
```

6. A

パラメータと引数の数の関係、および、可変パラメータに関する問題です。

関数に定義されたパラメータの数よりも引数の数が少ない場合、指定されなかったパラメータは未設定のまま処理が進められ、それぞれについて警告が発生します（**A**）。エラーとは異なり、警告が発生した場合、スクリプトの実行は続けられます。なお、パラメータの数よりも引数が多い場合は、警告は発生しません。

例 パラメータと引数の数

```
function func($x, $y) {
  // ...
}
func(1, 2, 3); // 3は無視されます
func(1, 2);    // 問題ありません
func(1);       // $yの引数がないため、警告が1つ発生します
func();        // $xと$yの引数がないため、警告が2つ発生します
```

関数が不特定の数のパラメータを必要とする場合は、**可変パラメータ**を用いることができます（B）。可変パラメータを扱うための関数には、func_num_args()、func_get_arg()、func_get_args()があります。

func_num_args()は、関数に与えられた引数の数を返します（C）。

func_get_arg()は、関数に与えられた引数のうち、指定された位置の引数を返します（D）。

書式

```
func_get_arg(int $arg_num)
$arg_num …… 位置
```

func_get_args()は、関数に与えられたすべての引数からなる配列を返します（D）。

以下は、func_num_args()とfunc_get_arg()を使用して、与えられた引数をすべてリスト（
～）として出力する関数の例です。

例 引数をリストとして出力（func_num_args()とfunc_get_arg()）

```
function print_list() {
  echo '<ul>' ;
  for ($i = 0; $i < func_num_args(); $i++) {
    echo '<li>' . func_get_arg($i) . '</li>';
  }
  echo '</ul>' ;
}
```

以下は上記の関数と同様の動作を行うコードですが、func_num_args()とfunc_get_arg()の代わりにfunc_get_args()を使用しています。

例 引数をリストとして出力（func_get_args()）

```
function print_list() {
  echo '<ul>' ;
  foreach (func_get_args() as $s) {
    echo '<li>' . $s . '</li>' ;
  }
  echo '</ul>' ;
}
```

7. B

関数の返り値に関する問題です。

関数は、処理結果である**返り値**を返すことができます。返り値はreturnキーワードの後ろに指定します（A）。

例 返り値の指定

```
function func($x, $y) {
  return $x + $y;
}

echo func(1, 2);
```

出力

```
3
```

返り値が不要である場合は、returnキーワードの後ろに何も書かず、「return;」とします。次の例は、文字列の最初の文字を表示する関数の例です。最初に、strlen()で文字列の長さを取得しています。文字列の長さが0の際は最初の文字を表示することはできないため、returnキーワードを用いて関数の呼び出しを終了させます。

例 returnキーワードを用いて関数の呼び出しを終了させる

```
function func($s) {
  if (strlen($s) == 0) {
    return;
  }
  echo '最初の文字: ', $s[0];
  return; // 省略可
}
func('abc');
```

出力

```
最初の文字: a
```

関数の末尾部分のreturnキーワードは、返り値がない場合は省略してもかまいません。

関数から複数の値を返すには、配列を使用します。「return 1, 2, 3;」のようにカンマ区切りで書く構文はありません（**B**）。

例 複数の返り値の指定

```
function func($x, $y) {
  return array($x + $y, $x * $y);
}

$result = func(1, 2);
echo '1 + 2 = ', $result[0], "\n";
echo '1 * 2 = ', $result[1], "\n";
```

出力

```
1 + 2 = 3
1 * 2 = 2
```

デフォルトでは、関数からは値のコピーが返されます（C）。関数名の前に&を付けることで、参照を返すことができます（D）。関数から返された参照を受け取るには、関数の呼び出し側でも関数名に&を付けます。次のコードを見てください。

例 参照を返す

```
function &func(&$x) {
  return $x;
}
$a = 1;
$b =& func($a);
$a = 2;
echo $b;
```

出力

```
2
```

func()の呼び出しにより、$bは$aに対する参照となります。つまり、$aと$bは同じ変数を指し示します。よって、$aに代入を行うと$bの出力が変化します。

8. C

可変関数と匿名関数に関する問題です。

可変関数とは、関数の名前を変数に代入したものです（A）。条件によって呼び出す関数を切り替えたい場合などに使用されます。可変関数は、その変数に( )を付けることで呼び出すことができます。

例 可変関数の呼び出し

```
$func = 'printf';
$func('hello');
```

出力

```
hello
```

可変関数を使う場合、指定された関数名に対応する関数が存在しなければ、コードの実行時にエラーが発生します（B）。関数名に対応する関数が定義されているかどうかは、以下のように**function_exists()**を使用して判定するこ

とができます。「関数名」で指定された関数が定義済みであればtrueを、そうでなければfalseを返します。

書式

```
function_exists(string $function_name)
$function_name…… 関数名
```

可変関数を使用する際、function_exists()によって関数が存在するかどうか確認してからその関数を呼び出すようにすると、コード実行時のエラーを回避することができます。

例 function_exists()による関数の存在確認

```
$func = 'printf';
if (function_exists($func)) {
  $func('hello');
} else {
  echo $func, 'は未定義です';
}
```

出力

```
hello
```

echo()やisset()は関数ではなく言語構造であるため、可変関数を用いて呼び出すことはできません（**C**）。次のようなコードはエラーとなります。

例 エラーになるコード

```
$func = 'echo';
$func('hello');
```

一時的に使用される関数が必要な場合は、通常の関数定義で定義した関数を使うほか、**匿名関数**を使用することもできます（D）。匿名関数には、ほかの関数と重複しないランダムな名前が割り振られます。そのため、プログラマが名前を考える必要がありません。

匿名関数は、**create_function()**を用いて作成します（E）。「パラメータリスト」で指定されたパラメータ、「コード」で指定された関数本体コードを持つ匿名関数を作り、ランダムな名前を割り当てて、その名前を返します。

書式

```
create_function(string $args, string $code)
$args …… パラメータリスト
$code …… コード
```

匿名関数は、変数に代入し、その変数に()を付けることで呼び出すことができます。

例 匿名関数の呼び出し

```
$func = create_function('$x', 'echo $x;');
$func('hello');
```

出力

```
hello
```

匿名関数は、array_filter()などの関数の引数として使用することができます。

例 array_filter()に匿名関数を指定

```
$func = create_function('$x', 'return $x >= 80;');
$a = array(78, 79, 80, 81, 82);
$b = array_filter($a, $func);
```

【作成される配列$b】

キー	2	3	4
値	80	81	82

第2章

文字列

■ 文字列の基本
■ 文字列の表示
■ 文字列の加工
■ エンコードとエスケープ
■ 文字列の比較
■ 文字列の操作と検索

☐ **1.** **PHPの文字列についての記述として、誤っているものを選択してください。（1つ選択）**

A. シングルクォート（'）でくくられた文字列と、ダブルクォート（"）でくくられた文字列では、使用できるエスケープシーケンスが異なる
B. 文字列中のバックスラッシュ（\）は、エスケープシーケンスを用いて「\\」と記述する
C. シングルクォート（'）でくくられた文字列の中に変数名を記述すると、変数の値が埋め込まれる
D. 複数行からなる文字列を簡潔に記述するには、ヒアドキュメントを使用する

➡ P35

☐ **2.** **文字列の表示についての記述として、誤っているものを選択してください。（1つ選択）**

A. print_r()とvar_dump()は、再帰的な構造を持つオブジェクトを表示することができる
B. strlen()を使用すると、文字列の長さを調べることができる
C. print_r()とvar_dump()は、与えられた値をわかりやすい形で出力する。この2つが出力する形式はまったく同じである
D. echoは、カンマ（,）区切りで複数の文字列を連続して出力することができる
E. echoとprint()は、関数ではなく言語構造であり、呼び出しの際の括弧を省略することができる

➡ P37

☐ **3.** **文字列関数についての記述として、誤っているものを選択してください。（1つ選択）**

A. trim()は、文字列の左右から指定文字を除去する
B. strtolower()は、文字列のすべての文字を小文字に変換する
C. substr()は、文字列から部分文字列を得る
D. strrev()は、文字列を逆順にしたものを返す
E. str_repeat()は、文字列があるパターンの繰り返しかどうかを調べる
F. str_pad()は、文字列にスペースまたは指定した文字列を継ぎ足す

➡ P40

4. エンコードを行う関数についての記述として、誤っているものを選択してください。（1つ選択）

A. htmlentities()は、HTMLの特別な文字、アクセント記号、ウムラウト付きの文字を、エンティティに置換する
B. htmlspecialchars()は、HTMLの「<」「>」「&」「"」といった文字をエンティティに置換するが、アクセント記号、ウムラウト付きの文字は置換しない
C. strip_tags()は、文字列中からHTMLタグで囲まれた部分を除去する。たとえば、「トップページに戻る」をstrip_tags()で処理すると、結果は「に戻る」となる
D. get_meta_tags()は、HTMLファイルからmetaタグの情報を取得する

➡ P43

5. エンコード処理およびエスケープ処理に関連する関数についての記述として、正しいものを選択してください。（1つ選択）

A. urlencode()は、空白文字を「+」に変換する
B. addslashes()は、SQLとして特別な意味を持つ文字の前にスラッシュ（/）を追加する
C. rawurlencode()は、空白文字を「+」に変換する

➡ P45

6. 文字列の比較についての記述として、誤っているものを選択してください。（1つ選択）

A. ==演算子は大文字・小文字を無視した比較を行うが、===演算子は大文字・小文字を区別した比較を行う
B. 大文字・小文字を無視して比較を行うには、strcasecmp()を使用する
C. 文字列の先頭から最初の数文字だけを比較するには、strncmp()を使用する
D. 文字列を「自然順」で比較するには、strnatcmp()を使用する

➡ P47

☐ **7**. 文字列の分割を行う関数についての記述として、誤っているものを選択してください。(1つ選択)

A. parse_url()は、URLをスキーム、ホスト、パスなどの構成要素に分解し、結果を連想配列として返す
B. implode()は、複数の文字列を結合した文字列を返す
C. strtok()は、文字列をトークンに分割し、結果を配列として返す
D. sscanf()は、書式文字列に従い、文字列を分割する
E. explode()は、文字列を区切り文字で分割し、結果を配列として返す

➡ P50

☐ **8**. 文字列の検索を行う関数についての記述として、誤っているものを選択してください。(1つ選択)

A. `strpos($x, $y)` は、文字列`$x`から、文字列`$y`が最初に現れる位置を検索し、見つかった位置を返す
B. `strstr($x, $y)` は、文字列`$x`から、文字列`$y`が最初に現れる位置を検索し、見つかった位置から末尾までの文字列を返す
C. `strrchr($x, $y)` は、文字列`$x`から、文字`$y`が最後に現れる位置を検索し、見つかった位置を返す
D. `strspn($x, $y)` は、文字列`$x`から、文字列`$y`に含まれる文字のみで構成される部分を調べ、その文字数を返す

➡ P52

第 2 章　文字列
解　答

1. C

PHPの**文字列**に関する基本的な知識の確認問題です。

PHPの文字列には、シングルクォート（'）でくくられた文字列と、ダブルクォート（"）でくくられた文字列の2種類があります。文字列の中に、シングルクォート、ダブルクォート、バックスラッシュ（\）などの特別な文字を置くには、**エスケープシーケンス**を使用します。エスケープシーケンスは、特別な文字の前にバックスラッシュを置く書き方です（B）。
なお、使用するフォントやOS環境によっては、バックスラッシュは「¥」として表示されることがあります。

シングルクォートでくくられた文字列と、ダブルクォートでくくられた文字列では、「\\」を除き、使用できるエスケープシーケンスが異なります（A）。

【シングルクォートでくくられた文字列で使用できるエスケープシーケンス】

文字	エスケープシーケンス
\	\\
'	\'

【ダブルクォートでくくられた文字列で使用できる主なエスケープシーケンス】

文字	エスケープシーケンス
\	\\
"	\"
$	\$
改行（LF）	\n
タブ	\t

例 エスケープシーケンスの使用例

```
echo '\\';
```

出力

```
\
```

※次ページに続く

ダブルクォートでくくられた文字列と、後述するヒアドキュメントでは、変数の展開が行われます。シングルクォートでくくられた文字列と、後述するNowdocでは、変数の展開は行われません（**C**）。**変数の展開**とは、文字列内に変数が記述された場合、変数の内容が文字列に埋め込まれる機能です。

例 変数の展開

```
$name = 'taro';
echo "Hello, $name!";
```

出力

```
Hello, taro!
```

展開を行いたい変数名の後ろに、識別子に使える文字が続くと、それが変数名の続きと解釈されてしまいます。次は、展開が失敗する例です。

例 変数の展開が失敗する例

```
$fruit = 'apple';
echo "There are 3 $fruits";
```

出力

```
There are 3
```

この場合は、変数名の部分を明確にするために、変数名を波括弧でくくり、{$fruit}と記述してください。

複数行からなる文字列を記述するには、**ヒアドキュメント**を使用します（D）。

例 ヒアドキュメントの使用例

```
echo '<p>';
echo <<<EOS
Hello,
PHP
World!
EOS;
echo '</p>';
```

出力

```
<p>Hello,
PHP
World!</p>
```

上記のコードの「<<<EOS」から「EOS;」までがヒアドキュメントです。「EOS」（End Of String）は**終端ID**と呼ばれるラベルで、PHPの識別子（変数名や関数名）の命名規則に従って自由に指定することができるので、ヒアドキュメント内に出現しないような文字列を選びます。
ヒアドキュメント内に記述された空白文字、改行、シングルクォート、ダブルクォートは、記述どおりに出力されます。ただし、最終行（上記の「World!」の行）の改行は取り除かれます。
ヒアドキュメントの末尾部分には、終端IDとセミコロン以外のものを書くことはできません。つまり、終端IDの前にスペースやタブを置いてインデントしたり、セミコロンの後ろにコメントを書いたりすることはできません。
PHP 5.3.0以降では、**Nowdoc**という機能が追加されました。Nowdocの構文は、「<<<」の右側に書く終端記号をシングルクォートでくくる以外は、ヒアドキュメントと同じです。Nowdocでは、変数の展開は行われません。

2. C

PHPの文字列出力に関する問題です。

文字列を出力するには、**echo**、**print()**、**printf()**を使用します。
echoとprint()は、関数ではなく**言語構造**です。言語構造は、関数と似たような使い方ができますが、関数とは異なる点もあるので注意してください。
echoとprint()は、呼び出しの際の、引数をくくる括弧を省略することができます（E）。

【echoとprint()の違い①】

コード	出力
echo('OK');	OK
echo 'OK';	OK
print('OK');	OK
print 'OK';	OK

echoは、複数の引数をカンマ（,）で区切って記述することが可能です（D）。print()には、そのような書き方はありません。

※次ページに続く

【echoとprint()の違い②】

コード	出力
`echo 'hello', 'world';`	`helloworld`
`print 'hello', 'world';`	（文法エラー）

echoで、カンマ区切りで複数の引数を記述する際、引数全体を括弧でくくると、文法エラーとなります。

【echoとprint()の違い③】

コード	出力
`echo 'hello', 'world';`	`helloworld`
`echo('hello', 'world');`	（文法エラー）

echoは返り値を持ちませんが、print()は返り値を持ちます（文字列が正しく表示された場合はtrueを返します）。したがって、print()は式の一部として記述することができます。

【echoとprint()の違い④】

コード	出力
`$disp = true;` `$disp ? echo 'OK' : echo 'NG';`	（文法エラー）
`$disp = true;` `$disp ? print 'OK' : print 'NG';`	`OK`

printf()を使用すると、**書式文字列**を用いて、値を所定の形式にフォーマットして出力することができます。

書式

```
printf(string $format [, mixed $args [, mixed $...]])
```

`$format` …… 書式文字列
`$args` …… 出力する式

printf()の書式文字列には、「**置換マーク**」を含めます。「置換マーク」は%で始まり、「d」や「s」などの「**型指定子**」で終わります。%と「型指定子」の間には、「03」や「-5」などの「**修飾子**」を指定することもできます。いくつか使用例を示します。

【printf()の使用例】

コード	出力	置換マークの意味
printf('%d', 12);	12	10進数（decimal）
printf('%3d', 12);	␣12	3桁（3）
printf('%03d', 12);	012	0埋め3桁（03）
printf('%f', 3.14);	3.140000	浮動小数点数（float）
printf('%1.2f', 3.14);	3.14	整数部1桁、小数部2桁
printf('%s', 'php');	php	文字列（string）
printf('%5s', 'php');	␣␣php	5桁（5）
printf('%-5s', 'php');	php␣␣	左寄せ5桁（-5）
printf('%s %d.%d', 'php', 5, 4);	php 5.4	対応する引数をそれぞれ出力
printf('%%');	%	%を出力

※ 表内の␣は空白文字を表す

デバッグ用に変数の内容をわかりやすい形で出力するには、**print_r()**や**var_dump()**を使用します。var_dump()を使用すると、NULLやfalseを文字列として表示することができ、また、それぞれの値の型の情報も表示されます。var_dump()では、複数の引数を「,」で区切って一度に指定することも可能です。次に、print_r()とvar_dump()の出力形式の違いをまとめます（**C**）。

【print_r()とvar_dump()】

値	print_r()による出力	var_dump()による出力
null		NULL
true	1	bool(true)
false		bool(false)
123	123	int(123)
'ok'	ok	string(2) "ok"

array(1, 2)	Array
	(
	　　[0] => 1
	　　[1] => 2
	)
		array(2) {
		　[0]=>
		　int(1)
		　[1]=>
		　int(2)
		}

class X {
　public $p = 1;
　private $q = 2;
}
new X()
	X Object
	(
	　[p] => 1
	　[q:X:private] => 2
	)
		object(X)#1 (2) {
		　["p"]=>
		　int(1)
		　["q":"X":private]=>
		　int(2)
		}

print_r()とvar_dump()は、オブジェクトの構造を再帰的にたどって出力を行います（A）。var_dump()を使用すると、自分自身への参照を含むオブジェクト

（$GLOBALSなど）であっても出力が可能です。

例 print_rの使用例

```
print_r($GLOBALS);
```

出力

```
Array
(
    [_GET] => Array
        (
        )

    [_POST] => Array
        (
        )

    [_COOKIE] => Array
        (
        )

    [_FILES] => Array
        (
        )

    [GLOBALS] => Array
 *RECURSION*
)
```

strlen()は、文字列の文字数を返します（B）。マルチバイト文字列（日本語など）の文字数を調べる場合は、**mb_strlen()**を使用します。

【strlen()とmb_strlen()】

コード	出力
echo strlen('hello');	5
echo mb_strlen('こんにちは');	5

3. E

PHPの基本的な文字列操作関数についての問題です。

trim()、**ltrim()**、**rtrim()**は、文字列から不要な文字を除去した文字列を返します（A）。第2引数の「除去文字」は省略可能で、省略した場合は空白／タブ／改行／キャリッジリターン／NULLバイト／垂直タブを除去します。これ

らの関数は、テキストボックスに入力された文字列から、余分な空白類を取り除く場合などに使用することができます。文字列の左右から、指定文字を除去する書式は次のとおりです。

書式

```
trim(string $str [, string $charlist])
$str…… 文字列
$charlist …… 取り除く文字
```

文字列の左端から指定文字を除去する書式は次のとおりです。

書式

```
ltrim(string $str [, string $charlist])
```

文字列の右端から指定文字を除去する書式は次のとおりです。

書式

```
rtrim(string $str [, string $charlist])
```

大文字・小文字を変換するには、**strtolower()**、**strtoupper()**、**ucfirst()**、**ucwords()**を使用します（B）。文字列のすべての文字を小文字に変換する書式は次のとおりです。

書式

```
strtolower(string $str)
$str…… 文字列
```

文字列のすべての文字を大文字に変換する書式は次のとおりです。

書式

```
strtoupper(string $str)
```

文字列の先頭の文字だけを大文字に変換する書式は次のとおりです。

書式

```
ucfirst(string $str)
```

文字列の各単語の先頭の文字を大文字に変換する書式は次のとおりです。

書式

```
ucwords(string $str)
```

substr()は、部分文字列のコピーを返します（C）。

書式

```
substr(string $string, int $start [, int $length])
$string …… 文字列
$start…… 開始位置
$length …… 文字数
```

文字列の「開始位置」で指定した文字位置から、「文字数」分をコピーして返します。第3引数の「文字数」を省略した場合は、「開始位置」から最後の文字までをコピーします。

文字列中に、特定のキーワードが何回登場するかを調べるには、**substr_count()**を使用します。

書式

```
substr_count(string $haystack, string $needle)
$haystack …… 文字列
$needle …… キーワード
```

substr_replace()は、文字列の置換を行います。

書式

```
substr_replace(mixed $string, mixed $replacement, mixed $start [, mixed $length])
$string …… 文字列1
$replacement …… 文字列2
$start…… 開始位置
$length …… 文字数
```

文字列1の「開始位置」から始まる「文字数」分の文字を、文字列2に置換したものを返します。

例 substr_replace()の使用例

```
echo substr_replace('opening', 'end', 0, 4);
```

出力

```
ending
```

strrev()は、文字列を逆順に置換します（D）。

書式

```
strrev(string $str)
$str …… 文字列
```

str_repeat()は、文字列を指定回数繰り返したものを作って返します（**E**）。

書式

```
str_repeat(string $input, int $multiplier)
$input…… 文字列
$multiplier …… 回数
```

str_pad()は、ある文字列に別の文字列を繰り返し加えます（F）。指定した長さになるまで文字列を追加する書式は次のとおりです。

書式

```
str_pad(string $input, int $pad_length [, string $pad_string = " "])
$input…… 文字列1
$pad_length …… 長さ
$pad_string …… 文字列2
```

指定された「長さ」になるまで、文字列1の右側に、文字列2を繰り返し追加したものを返します（F）。文字列2を省略した場合は空白文字が指定されたものとみなされます。

【str_pad()の使用例】

コード	出力
`echo str_pad('php', 10), '5.4';`	「php␣␣␣␣␣␣␣5.4」
`echo str_pad('php', 10, '.'), '5.4';`	「php.......5.4」
`echo str_pad('php', 10, '.␣'), '5.4'`	「php.␣.␣.␣.5.4」

※ 表内の␣は空白文字を表す

4. C

エンティティへの置換を行う関数と、HTMLコードやファイルを扱う関数に関する問題です。

HTMLコード中では、「<」「>」「&」「"」といった文字は特別な意味を持ちます。その特別な意味を打ち消して、これらの文字そのものをブラウザに表示させるには、それぞれ「<」「>」「&」「"」と記述する必要があります。これを**エンティティ**（文字参照）といいます。「<」「>」「&」「"」が含まれている文字列を、ページ内に表示されるテキストとして出力したい場

合は、これらをエンティティに置換する処理が必要になります。これを怠ると、出力したHTMLに思わぬ文字が紛れ込んで、ブラウザのページ表示を崩したり、セキュリティ上の問題点を生じさせたりすることになります。
htmlentities()や**htmlspecialchars()**を使用すると、文字列に含まれる特別な文字をエンティティに置換する処理を行うことができます。

書式

```
htmlentities(string $string [, int $flags = ENT_COMPAT
  [, string $encoding = 'ISO-8859-1' [, bool $double_encode = true]]])
$string ····· 入力文字列
$flags····· 置換オプション
$encoding ····· 文字エンコーディング
$double_encode ····· falseにすると既存のHTMLエンティティをエンコードしない
```

htmlentities()は、文字列に含まれる「<」「>」「&」「"」をそれぞれ「<」「>」「&」「"」に変換します。同時にアクセント記号やウムラウト付きの文字の置換も行います（A）。

書式

```
htmlspecialchars(string $string [, int $flags = ENT_COMPAT
  [, string $encoding = 'ISO-8859-1' [, bool $double_encode = true]]])
```

htmlspecialchars()の処理内容は、htmlentities()と同様ですが、アクセント記号やウムラウト付きの文字の置換は行いません（B）。これらの関数で使用される変換テーブルはget_translation_table()関数で取得できます。
htmlentities()とhtmlspecialchars()の第2、第3、第4引数は省略可能です。第2引数の「置換オプション」では、シングルクォートやダブルクォートの置換をするかどうかを指定することができます。省略時のデフォルトはENT_COMPATです。

【置換オプション】

置換オプション	「'」→「'」の置換	「"」→「"」の置換
ENT_COMPAT	しない	する
ENT_QUOTES	する	する
ENT_NOQUOTES	しない	しない

strip_tags()を使用すると、文字列からHTMLタグ、HTMLコメント、PHPタグを取り除くことができます。タグで囲まれたテキストはそのまま残ります。したがって、「トップページに戻る」をstrip_tags()で処理すると、結果は「トップページに戻る」となります（**C**）。この関

数は、たとえばユーザがテキストボックスに入力したテキストをページ内に出力する際に、テキスト中から余分なHTMLタグを取り除く場合に使用します。

書式

```
strip_tags(string $str [, string $allowable_tags])
$str ····· 文字列
$allowable_tags ····· 取り除かないタグ
```

「取り除かないタグ」は省略可能で、省略した場合はすべてのタグが削除されます。「取り除かないタグ」に指定したタグはそのまま残されます。

【strip_tags()の使用例】

コード	出力
`echo strip_tags('<i><b>php</b></i>');`	`php`
`echo strip_tags('<i><b>php</b></i>', '<b>');`	`<b>php</b>`

get_meta_tags()を使用すると、指定されたHTMLファイル内に記述されたmetaタグの情報を読み取り、その結果を連想配列として取得することができます（D）。

書式

```
get_meta_tags(string $filename)
$filename ····· ファイルパス
```

ファイルパスにはURLを指定することも可能です。

例 get_meta_tags()の使用例

```
$metainfo = get_meta_tags('index.html');
echo $metainfo['author'];
echo $metainfo['description'];
```

5. A

エンコード処理およびエスケープ処理を行う関数に関する問題です。
URLに文字列を含めて送信する際は、URLの仕様に沿って、送信文字列を適切に変換する必要があります。これを**エンコード**（符号化）といいます。エンコードされた文字列は、受信側で**デコード**（復号）を行って、元の文字列を取得します。

urlencode()や**rawurlencode()**は、**URLエンコード**を行う関数です。クエリ

文字列やクッキー値で使用される形式に沿ってエンコードを行う書式は次のとおりです。

書式

```
urlencode(string $str)
$str …… 文字列
```

RFC 3986形式に沿ってエンコードする書式は次のとおりです。

書式

```
rawurlencode(string $str)
```

urlencode()とrawurlencode()は、エンコード処理に違いがあります。たとえば、空白文字について、urlencode()は「+」、rawurlencode()は「%20」を返します（**A**、C）。

【urlencode()とrawurlencode()のエンコード処理の違い】

文字	urlencode()	rawurlencode()
アルファベット、ハイフン（-）、アンダースコア（_）、ピリオド（.）	変換しない	変換しない
チルダ（~）	%7E	変換しない
空白	+	%20
上記以外	%nn形式	%nn形式

urldecode()や**rawurldecode()**は、URLエンコードされた文字列のデコードを行います。urlencode()でエンコードされた文字列のデコードする書式は次のとおりです。

書式

```
urldecode(string $str)
```

rawurlencode()でエンコードされた文字列のデコードする書式は次のとおりです。

書式

```
rawurldecode(string $str)
```

URLエンコードと同様に、SQL文に文字列を含める際も、SQLの仕様に沿った変換（SQLとして特別な意味を持つ文字をエスケープする）が必要になります。

addslashes()と**stripslashes()**は、SQL文用のエンコードおよびデコードのために作成された関数ですが、後述の理由でSQL文で使用すべきではありませ

ん。addslashes()は、スラッシュではなくバックスラッシュを、シングルクォート（'）、ダブルクォート（"）、バックスラッシュ（\）、NULLバイトの前に付加します（B）。書式は次のとおりです。

書式

```
addslashes(string $str)
```

addslashes()でエンコードされた文字列をデコードする書式は次のとおりです。

書式

```
stripslashes(string $str)
```

ただし、ほぼすべてのデータベースシステムでaddslashes()による方法でのエスケープ処理は不適切です。したがって、SQL文のエスケープにaddslashes()を使用すべきではありません。使用するデータベースシステムに合わせて、専用のエスケープ関数を使用します。詳細は、第10章の解答3を参照してください。

6. A

文字列の比較に関する問題です。

文字列の比較を行うには、演算子を用いる方法と、関数を用いる方法の2通りがあります。
==演算子と**===演算子**は、2つの文字列の内容が同じかどうかを比較します。いずれの演算子でも、大文字・小文字を区別した比較を行います（**A**）。たとえば、「a」と「A」は異なる文字と判断されます。
次の表は、変数**$x**と**$y**に文字列を代入し、==演算子と===演算子で比較した結果です。

【文字列の比較】

$x	$y	$x == $y	$x === $y
'123'	'123'	true	true
'123abc'	'123'	false	false

==演算子で文字列と数値を比較する場合は注意が必要です。このとき、文字列が数値に**キャスト**（**型変換**）され、それから比較が行われます。一方、===演算子ではキャストが行われず、型が異なる2つの値を===演算子で比較するとfalseが返ります。次の表は、変数**$x**には文字を、変数**$y**には数値を代入し、==演算子と===演算子で比較した結果です。

【数値と文字列の比較】

$x	$y	$x == $y	$x === $y
'123'	123	true	false
'123abc'	123	true	false
'abc'	0	true	false
''	0	true	false

==演算子を用いると、思わぬデータが「等しい」と判定されてしまう場合があるため、値を==演算子でチェックする場合は注意が必要です。

文字列の辞書順における前後関係を比較するには、**<演算子**、**<=演算子**、**>演算子**、**>=演算子**を使用します。次の表は、変数$xと$yに文字列を代入し、さまざまな演算子で比較した結果です。

【<演算子、<=演算子、>演算子、>=演算子による文字列の比較】

$x	$y	$x == $y	$x < $y	$x <= $y	$x > $y	$x >= $y
'aa'	'aa'	true	false	true	false	true
'aa'	'ab'	false	true	true	false	false
'ba'	'aa'	false	false	false	true	true

文字列と数値を比較すると、文字列が数値にキャストされてから比較が行われます。次の例では、右辺を文字列とするか、数値とするかによって、結果が逆転しています。

【<演算子、<=演算子、>演算子、>=演算子による文字列と数値の比較】

$x	$y	$x == $y	$x < $y	$x <= $y	$x > $y	$x >= $y
'abc'	'123'	false	false	false	true	true
'abc'	123	false	true	true	false	false

文字列を比較する関数には、**strcmp()**や**strcasecmp()**があります。

strcmp()では、2つの文字列を比較し、辞書順の場合は負の数、等しい場合は0、それ以外の場合は正の数を返します。

書式

```
strcmp(string $str1, string $str2)
$str1 …… 文字列1
$str2 …… 文字列2
```

strcasecmp()では、はじめに2つの文字列を小文字に変換して比較するため、大文字・小文字の違いを無視して比較されます。そのほかは、strcmp()と同

様です（B）。

書式

```
strcasecmp(string $str1, string $str2)
```

2つの文字列の最初の数文字だけを比較したい場合は、**strncmp()**や**strncasecmp()**を使います（C）。
strncmp()では、先頭から指定された文字数までの文字列を比較します。そのほかはstrcmp()と同様です。

書式

```
strncmp(string $str1, string $str2, int $len)
$str1 …… 文字列1
$str2 …… 文字列2
$len…… 文字数
```

strncasecmp()では、先頭から指定された文字数までの文字列を大文字・小文字を無視して比較します。そのほかは、strcasecmp()と同様です。

書式

```
strncasecmp(string $str1, string $str2, int $len)
```

2つの文字列を「自然順」で比較したい場合は、**strnatcmp()**や**strnatcasecmp()**を使います（D）。自然順では、文字列に数値が含まれる場合、数値部分は数値の大小で比較されます。たとえば、「abc1」と「abc10」と「abc2」の3つを比較した際、通常の（文字列の辞書順の）比較では「abc1」＜「abc10」＜「abc2」となりますが、「自然順」で比較すると、「abc1」＜「abc2」＜「abc10」となります。

strnatcmp()では、文字列を自然順で比較します。そのほかはstrcmp()と同じです。

書式

```
strnatcmp(string $str1, string $str2)
$str1 …… 文字列1
$str2 …… 文字列2
```

strnatcasecmp()では、はじめに2つの文字列を小文字に変換してから比較します。そのほかはstrnatcmp()と同じです。

書式

```
strnatcasecmp(string $str1, string $str2)
```

7. C

文字列の分解・結合処理を行う関数に関する問題です。

explode()は、文字列を「区切り文字列」で分割し、配列を返します（E）。

書式

```
explode(string $delimiter, string $string)
$delimiter …… 区切り文字列
$string …… 文字列
```

例 explode()の使用例

```
$x = explode('-', 'php-5.4');
echo $x[1];
```

出力

```
5.4
```

implode()は、「区切り文字列」を間に入れながら、複数の文字列を結合します（B）。join()は、implode()のエイリアス（別名）であり、処理内容はimplode()と同じです。

書式

```
implode(string $glue, array $pieces)
または
join(string $glue, array $pieces)
$glue …… 区切り文字列
$pieces …… 文字列の配列
```

例 implode()の使用例

```
$x = array(1, 2, 3);
echo implode('-', $x);
```

出力

```
1-2-3
```

strtok()は、文字列をトークン（token、単語）に分割します。この関数を呼

び出すたびに、分割した新しいトークンを返します（**C**）。この関数は、最初のトークンを取得する際と、2番目以降のトークンを取得する際で、呼び出しの引数が変わります。

書式

```
strtok(string $str, string $token)
$str…… 文字列
$token…… 区切り文字
```

文字列を区切り文字で分割し、2番目以降のトークンを返す書式は次のとおりです。

書式

```
strtok(string $token)
$token…… 区切り文字
```

いずれも、トークンがない場合はfalseを返します。

例 strtok()の使用例

```
$x = 'hello,php,world';
$token = strtok($x, ',');
echo $token, '!';
while ($token = strtok(',')) {
  echo $token, '!';
}
```

出力

```
hello!php!world!
```

sscanf()は、printf()と同じ形式の「書式文字列」に沿って記述された文字列を分割し、値を読み取り、配列にして返します（D）。explode()やstrtok()よりも複雑な書式の文字列を分解することができます。文字列を書式文字列に従って解釈する書式は次のとおりです。

書式

```
sscanf(string $str, string $format)
$str…… 文字列
$format …… 書式文字列
```

「書式文字列」については、解答2を参照してください。

例 sscanf()の使用例

```
$x = '12/31 23:59';
$y = sscanf($x, '%d/%d %d:%d');
echo $y[0], '月';
echo $y[1], '日';
echo $y[2], '時';
echo $y[3], '分';
```

出力

```
12月31日23時59分
```

parse_url()は、URLをスキーム（「http」など）、ホスト、パスなどの構成要素に分解し、連想配列として返します（A）。

書式

```
parse_url(string $url)
$url …… URL
```

例 parse_url()の使用例

```
$url = 'http://www.php.net/downloads.php';
$info = parse_url($url);
echo 'scheme:', $info['scheme'], "\n";
echo 'host:', $info['host'], "\n";
echo 'path:', $info['path'], "\n";
```

出力

```
scheme:http
host:www.php.net
path:/downloads.php
```

8. C

文字列の検索に関する問題です。

strpos()は、文字列から、キーワードが最初に現れる位置（文字列の左端を0とする）を検索し、見つかった位置を返します（A）。

書式

```
strpos(string $haystack, mixed $needle)
$haystack …… 文字列
$needle …… キーワード
```

文字列内にキーワードが見つからない場合はfalseを返します。

例 strposの使用例

```
$x = 'linux,php,apache,php,mysql';
echo strpos($x, 'php');
```

出力

```
6
```

strrpos()は、文字列内でキーワードが最後に現れる位置を検索します。

書式

```
strrpos(string $haystack, mixed $needle)
```

文字列内にキーワードが見つからない場合はfalseを返します。

例 strrpos()の使用例

```
$x = 'linux,php,apache,php,mysql';
echo strrpos($x, 'php');
```

出力

```
17
```

strstr()は、strpos()と同様に、文字列から、キーワードが最初に現れる位置を検索しますが、見つかった位置ではなく、見つかった位置から末尾までの文字列を返します（B）。strchr()は、strstr()のエイリアス（別名）であり、処理内容はstrstr()と同じです。stristr()は、strstr()と同様の処理を行いますが、検索時に大文字・小文字の区別を無視します。

文字列内のキーワードから末尾までの文字列を返す書式は次のとおりです。

※次ページに続く

書式

```
strstr(string $haystack, mixed $needle)
または
strchr(string $haystack, mixed $needle)
```

文字列内のキーワード（大文字・小文字を無視）から末尾までの文字列を返す書式は次のとおりです。

書式

```
stristr(string $haystack, mixed $needle)
```

キーワードが見つからない場合はfalseを返します。

strrchr()は、文字列からキーワードが最後に現れる位置を検索し、見つかった場合は、そこから末尾までの文字列を返します（**C**）。

書式

```
strrchr(string $haystack, mixed $needle)
```

文字列内にキーワードが見つからない場合はfalseを返します。
検索文字に2文字以上の文字列を指定すると、最初の1文字だけが使用されます。strchr()とstrrchr()の使用例は次のとおりです。

例 strchr()とstrrchr()の使用例

```
echo strchr('abcabc', 'b'), "\n";
echo strrchr('abcabc', 'b'), "\n";
```

出力

```
bcabc
bc
```

機能別にまとめると以下のようになります。

【文字列内のキーワードの位置の検索】

	最初に現れる位置を検索	最後に現れる位置を検索
見つかった位置を返す	strpos()	strrpos()
見つかった位置から末尾までの文字列を返す	strstr() strchr() stristr()	strrchr()

strspn()は、文字列から、「検索文字列」に含まれる文字のみで構成される部分が何文字あるかを調べます（D）。

書式

```
strspn(string $subject, string $mask)
$subject ····· 文字列
$mask ····· 検索文字列
```

strspn()の使用例は次のとおりです。「検索文字列」に0〜9の数字を指定しているので、数字だけで構成される部分を先頭から探します。該当部分は「2012」となるので、strspn()は4を返します。

例 strspn()の使用例

```
$x = '2012/12/31';
echo strspn($x, '0123456789');
```

出力

```
4
```

strcspn()は、strspn()とは逆に、文字列から、「検索文字列」に含まれない文字のみで構成される部分が何文字目にあるかを調べます。

書式

```
strcspn(string $str1, string $str2)
$str1 ····· 文字列
$str2 ····· 検索文字列
```

strcspn()の使用例は次のとおりです。「検索文字列」にスペースとカンマを指定しているため、それらが含まれない部分を先頭から探します。該当部分は「Jan」となるので、strcspn()は3を返します。

例 strcspn()の使用例

```
$x = 'Jan 1, 2013';
echo strcspn($x, ' ,');
```

出力

```
3
```

第3章

配列

■ 配列の種類と作成

■ 配列の要素

■ 配列へのデータの格納

■ 多次元配列

■ 配列と変数との変換

■ 配列の順次処理

■ 配列のソート

■ 配列の集合演算

☐ **1.** **PHPの配列の特徴についての記述として誤っているものを選択してください。（1つ選択）**

A. PHPの配列には、インデックス配列と連想配列の2種類がある
B. 多次元配列を作成することができる
C. 重複するキーを持つ配列を作ることはできない
D. 配列のキーにはオブジェクトを指定することもできる

➡ P64

☐ **2.** **PHPの配列の特徴についての記述として正しいものを選択してください。（2つ選択）**

A. 配列$xの末尾に値を追加するには、「$x << 値」と書く
B. range()を使用すると、「1, 2, 3, 4, 5」や「a, b, c, d, e」といった、連続する値を持つ配列を簡単に作ることができる
C. length()を使用すると、配列の要素数を数えることができる
D. array_pad()は、配列が指定の長さになるように値を継ぎ足すことができる。たとえば、「1, 2」という配列が長さ5になるように、0を追加して「1, 2, 0, 0, 0」とすることができる

➡ P67

☐ **3.** **PHPの配列の展開と抽出についての記述として誤っているものを選択してください。（1つ選択）**

A. list()を使用すると、スクリプト内のすべての配列をリスト形式で出力することができる
B. array_slice()を使用すると、配列から要素を抽出することができる
C. array_chunk()を使用すると、配列を、指定した要素数の配列に分割することができる
D. array_splice()を使用すると、配列の不要な要素を削除したり、削除した位置に別の配列の要素を挿入したりすることができる

➡ P69

☐ **4.** PHPの配列のキー・値の操作についての記述として誤っているものを選択してください。（1つ選択）

A. array_keys()を使用すると、配列のキーだけを集めた配列を作ることができる
B. array_key_exists()を使用すると、配列に特定のキーが存在するかどうかを調べることができる
C. isset()は、array_key_exists()と同じように、値がNULLであるかどうかにかかわらず、配列に特定のキーが存在するかどうかを調べることができる
D. in_array()を使用すると、値が配列に存在するかどうかを調べることができる
E. array_search()を使用すると、値を指定して、その値に対応するキーを検索することができる

➡ P72

☐ **5.** PHPの配列を使用した変数の展開についての記述として誤っているものを選択してください。（1つ選択）

A. 配列が格納された変数でも、文字列内で変数の展開を行うことができる
B. 連想配列を使用して変数の展開を行う際に連想変数を波括弧でくくった場合は、キーをシングルクォートやダブルクォートでくくってはならない
C. 多次元配列を使用して、文字列内で変数の展開を行う場合は、配列全体を波括弧「｛」と「｝」でくくる

➡ P74

☐ **6.** PHPの配列と変数の変換についての記述として誤っているものを選択してください。（1つ選択）

A. extract()は、配列内のそれぞれのキーと値を、変数名とその値として、現在のスコープに変数を定義する
B. extract()で定義される変数には、オプションで任意の接頭辞を追加することができるため、既存の変数の上書きを避けることができる
C. compact()は、配列のデータを圧縮してコンパクトにし、メモリの消費量を削減することができる

➡ P75

☐ **7.** PHPの配列の順次処理についての記述として誤っているものを選択してください。（1つ選択）

A. foreach文を使用すると、配列の値を順に取り出して処理を行うことができるが、値に対応するキーは得られない
B. PHPの配列は「イテレータ」を持っており、これを用いて配列の要素を順に処理することができる
C. 配列がインデックス配列（キーが0から始まる整数である配列）の場合は、for文を使用して配列の要素を順に取り出すことができる

➡ P75

☐ **8.** PHPの配列の要素の処理についての記述として誤っているものを選択してください。（1つ選択）

A. array_walk()を使用すると、配列内のそれぞれの値について、それを引数として、指定したユーザ定義関数を呼び出すことができる
B. array_map()は、array_walk()のエイリアスである
C. array_reduce()は、ユーザ定義関数を用いて、配列内の各要素を1つの値に集約する
D. array_product()は、配列のすべての数の積を計算する
E. array_sum()は、配列のすべての数の和を計算する

➡ P79

☐ **9.** PHPの配列のソート（並べ替え）についての記述として誤っているものを選択してください。（1つ選択）

A. 値を基準とするソートを行うには、sort()、rsort()を使用する
B. 配列のキーを変更せずに、値を基準とするソートを行うには、asort()、arsort()を使用する
C. キーを基準とするソートを行うには、ksort()、krsort()を使用する
D. ユーザ定義関数を使用した任意の基準によるソートを行うには、usort()、uasort()、uksort()を使用する
E. usort()、uasort()、uksort()の第2引数に渡すユーザ定義関数では、第1引数と第2引数を比較して、trueまたはfalseを返さなければならない

➡ P81

10. PHPの配列のソート（並べ替え）についての記述として誤っているものを選択してください。（1つ選択）

A. natsort()と、natcasesort()は、配列の要素を自然順でソートする
B. array_multisort()は、複数の配列をマージしてからソートする
C. array_reverse()は、配列を逆順に並べ替える
D. array_flip()は、配列のキーと値を入れ替える
E. shuffle()は、配列をシャッフルする

➡ P83

11. PHPの配列の演算についての記述として誤っているものを選択してください。（1つ選択）

A. array_merge()は、複数の配列をマージする
B. array_diff()は、配列の差を計算する
C. array_filter()は、配列から、HTMLタグを含まない安全な値だけを集めた配列を求める

➡ P84

12. PHPの配列の集合演算についての記述として正しいものを選択してください。（1つ選択）

A. array_unique()は、ある配列の中に重複して含まれる値を集める
B. array_intersect()は、複数の配列に共通して含まれる値を集める
C. array_unique()が返す配列のキーは、元の配列のキーとは無関係に、0から始まる整数が順に割り振られたものとなる
D. array_intersect()が返す配列のキーは、元の配列のキーとは無関係に、0から始まる整数が順に割り振られたものとなる

➡ P87

13. PHPの配列への値の追加、配列からの値の削除についての記述として正しいものを選択してください。（2つ選択）

A. array_unshift()は、配列の先頭に値を追加する
B. array_shift()は、配列の先頭から値を削除する
C. array_push()は、配列の先頭に値を追加する
D. array_pop()は、配列の先頭から値を削除する

➡ P88

14. PHPの配列関数を使用した次のコードのうち、「AB」を出力するものを選択してください。（2つ選択）

A.
```
$x = array('X', 'A', 'B', 'C', 'D', 'E');
list(, $a ,$b) = $x;
echo $a, $b;
```

B.
```
$x = array('X', 'A', 'B', 'C', 'D', 'E');
array_slice($x, 2, 3);
echo $x[0], $x[1];
```

C.
```
$x = array('X', 'A', 'B', 'C', 'D', 'E');
$y = array_chunk($x, 2);
echo $y[0], $y[1];
```

D.
```
$x = array('X', 'A', 'B', 'C', 'D', 'E');
array_splice($x, 0, 2);
echo $x[0], $x[1];
```

E.
```
$x = array('name' => 'taro', 'age' => 20, 'blood' => 'AB');
extract($x, EXTR_PREFIX_ALL, 'person');
echo $person_blood;
```

➡ P89

15. 次のPHPのコードのうち、「php」を出力するものを選択してください。（1つ選択）

A.
```
$a['a'] = 'html';
$a[] = 'css';
$a[] = 'php';
echo $a[2];
```

B.
```
$a = array('html', 2 => 'css', 'php');
$a[3] = 'javascript';
echo $a[3];
```

C.
```
$a[1] = 'p';
$a[1.5] = 'h';
$a[2] = 'p';
foreach ($a as $v) {
  echo $v;
}
```

D.
```
$a = array();
array_push($a, 'h');
array_push($a, 'p');
array_unshift($a, 'p');
echo implode('', $a);
```

➡ P90

☐ **16.** **次のコードが「512346789」という文字列を出力するように、空欄①～③に入る記述の正しい組み合わせを選択してください。（1つ選択）**

```
function func($x, $y) {
  if ( ① ) {
    return -1;
  } else if ($y == 5) {
    ② ;
  } else {
    return $x - $y;
  }
}
$a = array(8, 2, 3, 6, 7, 9, 1, 5, 4);
③ ;
foreach ($a as $v) {
  echo $v;
}
```

A.
① $x == 5
② return 0
③ usort($a, 'func')

B.
① $x == 5
② return 1
③ usort($a, 'func')

C.
① $x == 5
② return 1
③ uksort($a, 'func')

D.
① $y != 5
② return 1
③ usort($a, 'func')

➡ P91

第3章　配列
解　答

1. D

PHPの配列に関する基本的な知識の確認問題です。

PHPの**配列**は、**キー**と**値**のペアが集まったものです。キーと値を対応付けて配列に格納したり、キーを指定して配列から値を取り出したりすることができます。キーと値のペアのことを**要素**と呼びます。

配列を作成するには、array()構文、または、角括弧構文を使用します。次に、**array()構文**を使用する例を示します。キーには、0から始まる連番が割り当てられます。

例 array()構文による配列の作成

```
array('a', 'b', 'c');
```

【作成される配列】

キー	0	1	2
値	'a'	'b'	'c'

array(**キー　=>　値**)と記述して、値のキーを明示的に指定することができます。なお、複数のキーと値を記述するときは、途中に改行を入れて揃えると読みやすくなります。

例 array()構文：キーを明示的に指定

```
array(1 => 'a',
      3 => 'b',
      5 => 'c');
```

【作成される配列】

キー	1	3	5
値	'a'	'b'	'c'

キーの指定を途中から省略することもできます。省略された部分には、明示的に指定されたキーから順に1ずつ増えるキーが割り当てられます。

例 array()構文：キーの指定を途中から省略

```
array(10 => 'a', 'b', 'c');
```

【作成される配列】

キー	10	11	12
値	'a'	'b'	'c'

角括弧構文を使用する例を次に示します。1行目の代入文の実行により、新しい配列が作成され、変数$aに格納されます。2、3行目の代入文の実行により、その配列の末尾（右端）にキーと値が追加されていきます。キーには、0から始まる整数が順に割り振られます。

例 角括弧構文による配列の作成

```
$a[] = 'a';
$a[] = 'b';
$a[] = 'c';
```

【変数$aに格納される配列】

キー	0	1	2
値	'a'	'b'	'c'

PHPの配列のキーと値は、内部的な並び順を持ちます。配列にキーと値を追加すると、値はキー順ではなく追加順に並びます。次のコードでは、角括弧構文を用い、キーを明示的に指定して値を追加しています。値が追加順に並んでいること、キーが0である値が存在しないことに着目してください。

例 角括弧構文：キーを明示的に指定

```
$b[3] = 'C';
$b[2] = 'B';
$b[1] = 'A';
```

【変数$bに格納される配列】

キー	3	2	1
値	'C'	'B'	'A'

PHPの配列には、インデックス配列と連想配列の2種類があります（A）。**インデックス配列**のキーは、0から始まる整数です。**連想配列**のキーは文字列です。次に、連想配列の作成例を2つ示します。