**判断题（5 \* 6）**

|  |  |  |
| --- | --- | --- |
|  | 在 PHP 中引用意味着用不同的名字访问同一个变量内容 | （ ） |
|  | 可以对PHP引用做指针运算 | （ ） |
|  | PHP引用是实际的内存地址 | （ ） |
|  | 在一个对象的方法中，$this 永远是调用它的对象的引用。 | （ ） |
|  | 当用 global $var 声明一个变量时实际上建立了一个到全局变量的引用 | （ ） |
|  | 当 unset 一个引用，只是断开了变量名和变量内容之间的绑定。 | （ ） |

**单选题（5 \* 15）**

 1．$a =& $b，关于$a和$b的描述，正确的是 ( )

|  |  |
| --- | --- |
| 1. $a和$b指向同一个变量 | 1. $a和$b指向不同的变量 |
| 1. $a指向 $b的地址 | 1. $b指向 $a的地址 |

 2．以下内容，不可以通过引用传递的是 ( )

|  |  |
| --- | --- |
| 1. 变量 | 1. new语句 |
| 1. 表达式 $a = 5 | 1. 从函数中返回的引用 |

3．执行下列代码后的结果是 ( )

1. <?php
2. $origin = 0;
3. $ref =& $origin;
4. $ref = 1;
5. echo $origin, $ref;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 01 | 1. 00 | 1. 10 | 1. 11 |

 4．执行下列代码后的结果是 ( )

1. <?php
2. **function** foo(&$var) { }
3. foo($a);
4. echo $a === null ? 'y' : 'n';
5. $b = [];
6. foo($b['b']);
7. echo array\_key\_exists('b', $b) ? 'y' : 'n';
8. $c = **new** StdClass;
9. foo($c->d);
10. echo property\_exists($c, 'd') ? 'y' : 'n';

|  |  |  |  |
| --- | --- | --- | --- |
| 1. yny | 1. ynn | 1. yyy | 1. nnn |

 5．执行下列代码后的结果是 ( )

1. <?php
2. $ref = 0;
3. $row =& $ref;
4. **foreach** (**array**(1, 2, 3) **as** $row) {
5. //nothing
6. }
7. echo $ref;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 0 | 1. 1 | 1. 2 | 1. 3 |

 6．执行下列代码后的结果是 ( )

1. <?php
2. **function** foo(&$var)
3. {
4. $var++;
5. }
6. $a=5;
7. foo($a);
8. echo $a;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 5 | 1. 6 | 1. 7 | 1. 8 |

 7．执行下列代码后的结果是 ( )

1. <?php
2. $a = 1;
3. $c = 2;
4. $b =& $a;
5. $a =& $c;
6. echo $a, " ", $b;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1 1 | 1. 1 2 | 1. 2 1 | 1. 2 2 |

 8．执行下列代码后的结果是 ( )

1. <?php
2. **function** foo(&$var)
3. {
4. $tmp = 'b';
5. $var =& $tmp;
6. }
7. $bar = 'a';
8. foo($bar);
9. echo $bar;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. a | 1. b | 1. ab | 1. ba |

 9．执行下列代码后的结果是 ( )

1. <?php
2. $a = 0;
3. $b =& $a;
4. unset($b);
5. echo isset($a) ? 'y' : 'n';
6. echo isset($b) ? 'y' : 'n';

|  |  |  |  |
| --- | --- | --- | --- |
| 1. yy | 1. yn | 1. ny | 1. nn |

 10．执行下列代码后的结果是 ( )

1. <?php
2. **class** foo {
3. **public** $value = 1;
4. **public** **function** &getValue() {
5. **return** $this->value;
6. }
7. }
8. $obj = **new** foo;
9. $value1 = $obj->getValue();
10. $value2 = &$obj->getValue();
11. $obj->value = 2;
12. echo $value1, $value2;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 11 | 1. 12 | 1. 21 | 1. 22 |

 11．执行下列代码后的结果是 ( )

1. <?php
2. $GLOBALS["var"] = 1;
3. **function** foo() {
4. **global** $var;
5. unset($var);
6. }
7. foo();
8. echo isset($GLOBALS["var"]) ? 'y' : 'n';

|  |  |  |  |
| --- | --- | --- | --- |
| 1. y | 1. n | 1. yy | 1. nn |

12．执行下列代码后的结果是 ( )

1. <?php
2. **class** My{
3. **public** $value;
4. **function** get(&$ref){
5. $ref = $this;
6. }
7. }
8. $m = **new** My();
9. $m->value = 'a';
10. $m->get($m2);
11. $m2->value = 'b';
12. echo $m->value,$m2->value;

|  |  |  |  |
| --- | --- | --- | --- |
| 1. aa | 1. ab | 1. ba | 1. bb |

 \*13．执行下列代码后的结果是 ( )

1. <?php
2. **function** foo(&$bar) {
3. $bar = "hello ";
4. }
5. foo($unset);
6. echo($unset);
7. foo($set = "set");
8. echo($set);

|  |  |  |  |
| --- | --- | --- | --- |
| 1. hello set | 1. set | 1. hello | 1. hello hello |

 \*14．执行下列代码后的结果是 ( )

1. <?php
2. $arr3 = **array**(1);
3. $a =& $arr3[0];
4. $arr4 = $arr3;
5. $arr4[0]++;
6. echo "$a ";
7. echo "{$arr3[0]} ";
8. echo "{$arr4[0]}";

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1 1 1 | 1. 1 1 2 | 1. 1 2 2 | 1. 2 2 2 |

\*15．执行下列代码后的结果是 ( )

1. <?php
2. $arr = ['1', '2', '3', '4'];
3. **foreach** ($arr **as** &$i) {}
4. **foreach** ($arr **as** $i) {}
5. echo implode($arr, ', ');

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 1,1,1,1 | 1. 1,2,3,3 | 1. 1,2,3,4 | 1. 4,4,4,4 |