

Введение.....	2
Задача №1	3
Условие:.....	3
Решение:	3
Задача №2.....	5
Условие:.....	5
Решение:	5
Задача №3.....	13
Условие:.....	13
Решение:	13
Задача №4.....	15
Условие:.....	15
Решение:	15

Введение

Для решения заданий использовался следующий стек технологий:

- Windows 10
- Apache-2.4
- PHP-5.6;
- MySQL-5.6
- PHPMysqlAdmin

Исходной код всех решений представлен в созданном репозитории git и выложен на github.com: <https://github.com/lmakarenko/tek.torg.test>

Задача №1

Условие:

Написать функцию, которая на входе принимает строку из скобок, и возвращает true если все открытые скобки закрыты, иначе - false. Возможные варианты скобок: ()[]{}

Пример:

```
"(){}[]" => true
"([{}])" => true
"()" => false
"[]" => false
"[({})]()" => false
```

Решение:

Для решения данного задания используется созданный класс Task1\strChecker, содержащий бизнес-логику. При обработке строки происходит проход по всем символам строки, формирование стэка (массив объектов, формирующийся по правилу LIFO) из открытых скобок, и проверка на каждой итерации цикла текущего символа строки. В случае, если после открытой скобки следует недопустимый символ (символ закрытия скобки другого типа, нежели чем предыдущая открытая скобка), то выбрасывается исключение с информацией о строке, недопустимом символе и его позиции в строке (рисунок 1).

Полный код решения содержит комментарии и представлен на github: <https://github.com/lmakarenko/tek.torg.test/tree/master/task1>

Результат работы решения для заданного в условии набора данных представлен на рисунке 1.

Проверка строки : (){}[]

Успех

Проверка строки : ([{}])

Успех

Проверка строки : {}

Неверная скобка в строке: {}, символ '}' в позиции 1

Проверка строки : [()]

Неверная скобка в строке: [()], символ ']' в позиции 2

Проверка строки : [({})][]

Неверная скобка в строке: [({})][], символ ']' в позиции 7

Рисунок 1 – Результат работы решения задачи №1

Задача №2

Условие:

Написать SQL-запрос. Вывести данные по ученикам в формате: age, gender, is_excellent, total. Отсортировать по age и is_excellent в обратном порядке, вывести только если total > 1.

Здесь:

is_excellent - является ли отличником (0 - не является, 1 - является);

total - количество записей (выводить при значении 1 и более).

```
CREATE TABLE `users` (  
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
  `age` int(11) DEFAULT NULL,  
  `gender` tinyint(1) DEFAULT '0' COMMENT '0 - male, 1 - female',  
  `mark` int(11) DEFAULT NULL COMMENT '1-5',  
  PRIMARY KEY (`id`)  
);
```

Решение:

В данном случае БД не нормализована, информация об учениках и их оценках хранится в одной таблице, при этом в условии задачи требуется выводить вычисляемое поле total, что не имеет смысла, поскольку в таблице users каждая запись ученика уникальна (первичный ключ ID), и поле total будет всегда содержать 1. Тем не менее для выборки данных из представленной таблицы, учитывая ряд заданных условий, и полагая, что значение поля mark равное NULL является признаком нулевого кол-ва записей (total = 0), можно воспользоваться SQL-запросом (листинг 2)¹:

Листинг 2 – SQL-запрос выборки данных

```
SELECT  
  t.age,  
  t.gender,
```

¹ В качестве СУБД при решении задачи использовалась MySQL-5.6

```

t.mark,
IF(t.mark = 5, 1, 0) as is_excellent,
1 as total
FROM
`users-old` as t
WHERE
t.mark IS NOT NULL
ORDER BY
t.`age` DESC, `is_excellent` DESC

```

Результат выполнения SQL-запроса в среде PHPMyAdmin представлен на рисунке 2:

age ▼ 1	gender 0 - male, 1 - female	is_excellent ▼ 2
33	0	1
32	0	1
30	1	1
30	1	0
29	1	0
26	1	0
22	1	1
22	0	0
20	1	1
18	1	0

Рисунок 2 – Результат выполнения SQL-запроса в PHPMyAdmin

Однако можно рассмотреть вариант с нормализацией представленной БД, и разбиением таблицы на несколько связанных между собой таблиц. Для нормализации, в самом простом случае, можно разбить данную таблицу на 2 таблицы, следующим образом (таблица 1):

Таблица 1 – Список таблиц БД после нормализации

Имя таблицы	Описание
users	<p>Содержит информацию о пользователях:</p> <ul style="list-style-type: none"> • идентификатор пользователя (первичный ключ); • возраст; • пол.

users_marks	<p>Содержит информацию об оценках каждого пользователя:</p> <ul style="list-style-type: none"> • идентификатор записи (первичный ключ); • идентификатор пользователя (внешний ключ); • оценка; • временная метка выставления оценки.
-------------	--

Схема новой БД будет выглядеть следующим образом (рисунок 3):



Рисунок 3 – Схема нормализованной БД в PHPMysqlAdmin

Как можно заметить, таблица users связана отношением один-ко-многим с таблицей users_marks.

SQL-дамп нормализованной БД представлен в листинге 1:

```
--
-- Структура таблицы `users`
--

CREATE TABLE `users` (
  `user_id` int(11) UNSIGNED NOT NULL,
  `age` int(11) DEFAULT NULL,
  `gender` tinyint(1) DEFAULT '0' COMMENT '0 - male, 1 - female'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `users`
--

INSERT INTO `users` (`user_id`, `age`, `gender`) VALUES
(1, 18, 1),
(2, 20, 1),
```

```
(3, 22, 0),
(4, 22, 1),
(5, 30, 1),
(6, 32, 0),
(7, 30, 1),
(8, 33, 0),
(9, 26, 1),
(10, 29, 1),
(11, 18, 0);
```

```
-- -----
```

```
--
```

```
-- Структура таблицы `users_marks`
```

```
--
```

```
CREATE TABLE `users_marks` (
  `id` int(11) UNSIGNED NOT NULL,
  `user_id` int(11) UNSIGNED NOT NULL,
  `mark` int(11) UNSIGNED NOT NULL COMMENT '1-5',
  `tms` timestamp NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
--
```

```
-- Дамп данных таблицы `users_marks`
```

```
--
```

```
INSERT INTO `users_marks` (`id`, `user_id`, `mark`, `tms`) VALUES
(1, 1, 5, '2019-03-28 14:20:23'),
(2, 2, 3, '2019-03-28 14:20:23'),
(3, 3, 4, '2019-03-28 14:20:23'),
(4, 4, 4, '2019-03-28 14:20:23'),
(5, 5, 3, '2019-03-28 14:20:23'),
(6, 6, 4, '2019-03-28 14:20:23'),
(7, 7, 5, '2019-03-28 14:20:23'),
(8, 8, 2, '2019-03-28 14:20:23'),
(9, 9, 5, '2019-03-28 14:20:23'),
(10, 10, 4, '2019-03-28 14:20:23'),
(11, 1, 5, '2019-03-28 14:20:23'),
(12, 5, 4, '2019-03-28 14:20:23');
```

```
--
```

```
-- Индексы сохранённых таблиц
```

```
--
```



```

--
-- Индексы таблицы `users`
--
ALTER TABLE `users`
  ADD PRIMARY KEY (`user_id`);

--
-- Индексы таблицы `users_marks`
--
ALTER TABLE `users_marks`
  ADD PRIMARY KEY (`id`),
  ADD KEY `FK_users` (`user_id`);

--
-- AUTO_INCREMENT для сохранённых таблиц
--

--
-- AUTO_INCREMENT для таблицы `users`
--
ALTER TABLE `users`
  MODIFY `user_id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=12;

--
-- AUTO_INCREMENT для таблицы `users_marks`
--
ALTER TABLE `users_marks`
  MODIFY `id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;

--
-- Ограничения внешнего ключа сохраненных таблиц
--

--
-- Ограничения внешнего ключа таблицы `users_marks`
--
ALTER TABLE `users_marks`
  ADD CONSTRAINT `FK_users` FOREIGN KEY (`user_id`) REFERENCES `users`
  (`user_id`) ON DELETE CASCADE ON UPDATE CASCADE;
COMMIT;

```

Заполним тестовыми данными созданные таблицы (рисунки 4-5):

+ Параметры

				user_id	age	gender 0 - male, 1 - female
<input type="checkbox"/>				1	18	1
<input type="checkbox"/>				2	20	1
<input type="checkbox"/>				3	22	0
<input type="checkbox"/>				4	22	1
<input type="checkbox"/>				5	30	1
<input type="checkbox"/>				6	32	0
<input type="checkbox"/>				7	30	1
<input type="checkbox"/>				8	33	0
<input type="checkbox"/>				9	26	1
<input type="checkbox"/>				10	29	1
<input type="checkbox"/>				11	18	0

☐ Отметить все
 С отмеченными:

Рисунок 4 – Данные таблицы users в RHPMyAdmin

				id	user_id	mark 1-5	tms
<input type="checkbox"/>				1	1	5	2019-03-28 19:20:23
<input type="checkbox"/>				2	2	3	2019-03-28 19:20:23
<input type="checkbox"/>				3	3	4	2019-03-28 19:20:23
<input type="checkbox"/>				4	4	4	2019-03-28 19:20:23
<input type="checkbox"/>				5	5	3	2019-03-28 19:20:23
<input type="checkbox"/>				6	6	4	2019-03-28 19:20:23
<input type="checkbox"/>				7	7	5	2019-03-28 19:20:23
<input type="checkbox"/>				8	8	2	2019-03-28 19:20:23
<input type="checkbox"/>				9	9	5	2019-03-28 19:20:23
<input type="checkbox"/>				10	10	4	2019-03-28 19:20:23
<input type="checkbox"/>				11	1	5	2019-03-28 19:20:23
<input type="checkbox"/>				12	5	4	2019-03-28 19:20:23

☐ Отметить все
 С отмеченными:

Рисунок 5 – Данные таблицы users_marks в RHPMyAdmin

После нормализации, можно сформировать запрос на выборку записей с указанными в условии задачи критериями, при этом также можно учесть число имеющихся оценок (total) для каждого пользователя (число записей для пользователя в таблице users_marks), и если $total \geq 1$, то выводить информацию по данному пользователю (INNER JOIN).

Пример кода рассматриваемого SQL-запроса представлен в листинге 3:

Листинг 3 – SQL-запрос выборки данных для нормализованный БД

```
SELECT
-- идентификатор ученика (выборка уникальных значений)
DISTINCT u.user_id,
-- возраст ученика
u.age,
-- пол ученика
u.gender,
-- признак того, что ученик отличник - подзапрос, подсчитывающий число записей
из таблицы оценок для данного пользователя, у которых оценка не равна 5, т.е.
для отличников данный подзапрос вернет 0
IF((SELECT COUNT(id) FROM users_marks WHERE user_id = u.user_id AND mark != 5)
= 0, 1, 0) is_excellent,
-- кол-во записей для конкретного ученика в таблице оценок users_marks
(SELECT COUNT(id) FROM users_marks WHERE user_id = u.user_id) total
FROM
users u
-- внутренний join с таблицей оценок по полю user_id
INNER JOIN users_marks um ON um.user_id = u.user_id
-- сортировка по убыванию для полей age, is_excellent
ORDER BY
u.age DESC, is_excellent DESC
```

Результат выполнения данного SQL-запроса в среде PHPMыAdmin представлен на рисунке 6.

user_id	age ▾ 1	gender	is_excellent ▾ 2	total
8	33	0	0	1
6	32	0	0	1
7	30	1	1	1
5	30	1	0	2
10	29	1	0	1
9	26	1	1	1
3	22	0	0	1
4	22	1	0	1
2	20	1	0	1
1	18	1	1	2

Рисунок 6 - Результат выполнения SQL-запроса на выборку из нескольких таблиц в PHPMыAdmin

Как можно заметить, записи для пользователя с `user_id = 11` нет в выборке, поскольку для данного пользователя нет записей в таблице оценок `users_marks`, что позволяет выбирать только те записи, у которых `total >= 1`.

Также полученный запрос можно инкапсулировать в представление для дальнейшего использования (например, материализованное представление для ускорения выполнения запроса).

Задача №3

Условие:

Убрать и упростить лишние конструкции и выражения так, чтобы логика работы функции не поменялась:

```
function c($a) {  
    $a = ($a & 0x3f) + 1.9 >> 0;  
    $c = !1 !== !!$a ? --$a : $a++ ? ~~$a : $a++;  
    if (!$c == false) {  
        for ($a --= ($c * 3), $c = $a; $a < $c << 0b10;) {  
            $a *= $c;  
        }  
        return 0 | (int)$a;  
    } else {  
        return 0 | sqrt($c);  
    }  
}
```

Решение:

Код функции был упрощен, и инкапсулирован в тело новой функции Task3\c1. После чего был выведен результат работы обеих функций для сравнения.

Полный код решения содержит комментарии и представлен на github: <https://github.com/lmakarenko/tek.torg.test/tree/master/task3>

Результат работы решения для тестового набора данных представлен на рисунке 7. Как можно заметить, значения функций, для заданного диапазона аргументов, одинаковые.

$$c(-100) = 12544, c1(-100) = 12544$$

$$c(-90) = 23104, c1(-90) = 23104$$

$$c(-80) = 36864, c1(-80) = 36864$$

$$c(-70) = 53824, c1(-70) = 53824$$

$$c(-60) = 256, c1(-60) = 256$$

$$c(-50) = 3136, c1(-50) = 3136$$

$$c(-40) = 9216, c1(-40) = 9216$$

$$c(-30) = 18496, c1(-30) = 18496$$

$$c(-20) = 30976, c1(-20) = 30976$$

$$c(-10) = 46656, c1(-10) = 46656$$

$$c(0) = 0, c1(0) = 0$$

$$c(10) = 1600, c1(10) = 1600$$

$$c(20) = 6400, c1(20) = 6400$$

$$c(30) = 14400, c1(30) = 14400$$

$$c(40) = 25600, c1(40) = 25600$$

$$c(50) = 40000, c1(50) = 40000$$

$$c(60) = 57600, c1(60) = 57600$$

$$c(70) = 576, c1(70) = 576$$

$$c(80) = 4096, c1(80) = 4096$$

$$c(90) = 10816, c1(90) = 10816$$

Рисунок 7 - Результат работы решения задачи №3

Задача №4

Условие:

Написать функцию, которая принимает в качестве аргументов число (количество строк в итоговом массиве) и символ (для заполнения массива, см. пример), возвращает массив строк.

Пример:

```
build(6, "*")
[
  '      *      ',
  '     ***     ',
  '    *****  ',
  '   *********  ',
  '  ************ ',
  ' *****',
  '*****'
]
```

Решение:

Для решения данного задания используется созданный класс Task4\treeBuilder, содержащий бизнес-логику.

Полный код решения содержит комментарии и представлен на github: <https://github.com/lmakarenko/tek.torg.test/tree/master/task4>

Результат работы решения для тестового набора данных представлен на рисунке 8.

Построение дерева : число строк 1, символ '0'

Неверное число строк (1), число строк должно быть ≥ 2

Построение дерева : число строк 2, символ '0'

000

Успех

Построение дерева : число строк 30, символ '*'

[illegible]

Рисунок 8 - Результат работы решения задачи №4