Решение необходимо оформить в виде ссылки на Google Doc. В него нужно собрать все ответы и ссылки.

## Задание 1:

**Задание:** Спроектировать схему БД для хранения библиотеки. Интересуют авторы и книги.

**Дополнительное задание:** Написать SQL который вернет список книг, написанный 3-мя соавторами. Результат: книга - количество соавторов.
*Решение должно быть представлено в виде ссылки на sqlfiddle.com.*

## Задание 2:

**Задание:** Реализовать счетчик вызова скрипта. Было принято решение, хранить данные в файле.

```php
<?php

file_put_contents("./counter.txt", file_get_contents("./counter.txt") + 1);
```

**Вопрос:** Какие проблемы имеет данные подход? Как вы их можете решить?
(Нельзя использовать другие технологии)

**Дополнительный вопрос:** Через некоторое время нагрузка на сервер значительно выросла. Какие проблемы вы видите? Как вы их можете решить?
Если бы вы могли выбрать другую технологию, то какую и почему?

## Задание 3:

**Задание:** Проведите Code Review. Необходимо написать, с чем вы не согласны и почему.

**Дополнительное задание:** Напишите свой вариант.
*Решение должно быть представлено в виде ссылки на codeshare.io.*

**Требования были:** Добавить возможность получения данных от стороннего сервиса.

```php
<?php

namespace src\Integration;

class DataProvider
{
    private $host;
    private $user;
    private $password;

    /**
     * @param $host
     * @param $user
     * @param $password
     */
    public function __construct($host, $user, $password)
    {
        $this->host = $host;
        $this->user = $user;
        $this->password = $password;
    }

    /**
     * @param array $request
     *
     * @return array
     */
    public function get(array $request)
    {
        // returns a response from external service
    }
}
<?php

namespace src\Decorator;

use DateTime;
use Exception;
use Psr\Cache\CacheItemPoolInterface;
use Psr\Log\LoggerInterface;
use src\Integration\DataProvider;

class DecoratorManager extends DataProvider
{
    public $cache;
    public $logger;

    /**
```

```
     * @param string $host
     * @param string $user
     * @param string $password
     * @param CacheItemPoolInterface $cache
     */
    public function __construct($host, $user, $password, CacheItemPoolInterface
$cache)
    {
        parent::__construct($host, $user, $password);
        $this->cache = $cache;
    }

    public function setLogger(LoggerInterface $logger)
    {
        $this->logger = $logger;
    }

    /**
     * {@inheritdoc}
     */
    public function getResponse(array $input)
    {
        try {
            $cacheKey = $this->getCacheKey($input);
            $cacheItem = $this->cache->getItem($cacheKey);
            if ($cacheItem->isHit()) {
                return $cacheItem->get();
            }

            $result = parent::get($input);

            $cacheItem
                ->set($result)
                ->expiresAt(
                    (new DateTime())->modify('+1 day')
                );

            return $result;
        } catch (Exception $e) {
            $this->logger->critical('Error');
        }

        return [];
    }

    public function getCacheKey(array $input)
    {
        return json_encode($input);
```

```
    }
}
```

## Задание 4:

У вас нет доступа к библиотекам для работы с большими числами. Дано два числа в виде строки. Числа могут быть очень большими, могут не поместиться в 64 битный integer.

**Задание:** Написать функцию которая вернет сумму этих чисел.
*Решение должно быть представлено в виде ссылки на codeshare.io.*

## Задание 5:

**Дано:**

```
CREATE TABLE test (
  id INT NOT NULL PRIMARY KEY
);

INSERT INTO test (id) VALUES (1), (2), (3), (6), (8), (9), (12);
```

**Задание:** Написать SQL запрос который выведет все пропуски.

**Результат:**

```
FROM | TO
3    | 6
6    | 8
9    | 12
```

*Решение должно быть представлено в виде ссылки на sqlfiddle.com.*