Задание 1 (на таймеры)

- Написать функцию `counter(n)`, которая выводит в консоль раз в секунду числа n, n-1 ... 2, 1, 0 и останавливается.
- Написать функцию `createCounter(n)`, возвращающую объект с методами:
  - start() -- запускает (или возобновляет) счётчик с интервалом 1 секунда: N, N-1.
  - pause() -- приостанавливает счёт, но не сбрасывает счётчик.
  - stop() -- останавливает счёт, сбрасывает счётчик.

```
// Пример использования функции counter
counter(5);

// Пример использования функции createCounter
const myCounter = createCounter(5);
myCounter.start(); // Запускает счетчик
// Пауза счетчика через 3 секунды
setTimeout(() => {
    myCounter.pause();
}, 3000);
// Запуск счетчика после паузы через 2 секунды
setTimeout(() => {
    myCounter.start();
}, 5000);
// Остановка счетчика через 10 секунд
setTimeout(() => {
    myCounter.stop();
}, 10000);
```

```
function counter(n) {
    let current = n;
    const intervalId = setInterval(() => {
        console.log(current);
        if (current === 0) {
            clearInterval(intervalId);
        } else {
            current--;
        }
    }, 1000);
}

function createCounter(n) {
    let current = n;
    let intervalId;

    return {
        start() {
            intervalId = setInterval(() => {
                console.log(current);
                if (current === 0) {
                    clearInterval(intervalId);
                } else {
                    current--;
                }
            }, 1000);
        },
        pause() {
            clearInterval(intervalId);
        },
        stop() {
            clearInterval(intervalId);
            current = n;
        }
    };
}
```

Задание 2 (на промисы)

- Написать функцию delay(N), возвращающую промис, который сделает resolve() через N секунд.
- Решить задачу со счётчиком `N, N-1 ... 2, 1, 0` через функцию `delay`.
- Написать функцию, возвращающую название первого репозитория на github.com по имени пользователя (2 последовательных запроса: https://api.github.com/users/%USERNAME%).

```
function delay(N) {
    return new Promise(resolve => {
        setTimeout(() => {
            resolve();
        }, N * 1000);
    });
}

// Функция для выполнения счетчика
async function countDown(N) {
    for (let i = N; i >= 0; i--) {
        console.log(i);
        await delay(1);
    }
}

// Функция для получения названия первого репозитория пользователя на GitHub
async function getFirstRepo(username) {
    const response = await fetch(`https://api.github.com/users/${username}`);
    const userData = await response.json();
    const reposUrl = userData.repos_url;
    const reposResponse = await fetch(reposUrl);
    const reposData = await reposResponse.json();
    if (reposData.length > 0) {
        return reposData[0].name;
    } else {
        return "Пользователь не имеет репозиториев.";
    }
}

// Пример использования функции countDown
countDown(5);

// Пример использования функции getFirstRepo
getFirstRepo('octocat').then(repoName => {
    console.log(`Первый репозиторий пользователя: ${repoName}`);
}).catch(error => {
    console.error('Ошибка при получении данных:', error);
});
```

Задание 3 (на async/await)

Перепишите, используя async/await вместо .then/catch.

В функции getGithubUser замените рекурсию на цикл, используя async/await.

```
class HttpError extends Error {
  constructor(response) {
    super(`${response.status} for ${response.url}`);
    this.name = 'HttpError';
    this.response = response;
  }
}

function loadJson(url) {
  return fetch(url)
    .then(response => {
      if (response.status == 200) {
        return response.json();
      } else {
        throw new HttpError(response);
      }
    })
}

// Запрашивается логин, пока github не вернёт существующего
пользователя.
function getGithubUser() {
  let name = prompt("Введите логин?", "iliakan");

  return loadJson(`https://api.github.com/users/${name}`)
    .then(user => {
      alert(`Полное имя: ${user.name}.`);
      return user;
    })
    .catch(err => {
      if (err instanceof HttpError && err.response.status ==
404) {
        alert("Такого пользователя не существует, пожалуйста,
повторите ввод.");
        return demoGithubUser();
      } else {
        throw err;
      }
    });
}

getGithubUser();
```

```
class HttpError extends Error {
  constructor(response) {
    super(`${response.status} for ${response.url}`);
    this.name = 'HttpError';
    this.response = response;
  }
}

async function loadJson(url) {
  const response = await fetch(url);
  if (response.status === 200) {
    return response.json();
  } else {
    throw new HttpError(response);
  }
}

// Запрашивается логин, пока github не вернёт существующего пользователя.
async function getGithubUser() {
  let user;
  while (!user) {
    let name = prompt("Введите логин?", "iliakan");
    try {
      user = await loadJson(`https://api.github.com/users/${name}`);
      alert(`Полное имя: ${user.name}.`);
    } catch (err) {
      if (err instanceof HttpError && err.response.status == 404) {
        alert("Такого пользователя не существует, пожалуйста, повторите ввод.");
      } else {
        throw err;
      }
    }
  }
  return user;
}

// Вызов функции getGithubUser с помощью async/await
(async () => {
  try {
    const githubUser = await getGithubUser();
    console.log(githubUser);
  } catch (error) {
    console.error('Ошибка:', error);
  }
})();
```