Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 4
Взаимодействие серверов и знакомство с языком Prolog

Студент Жигалкин Д.Р

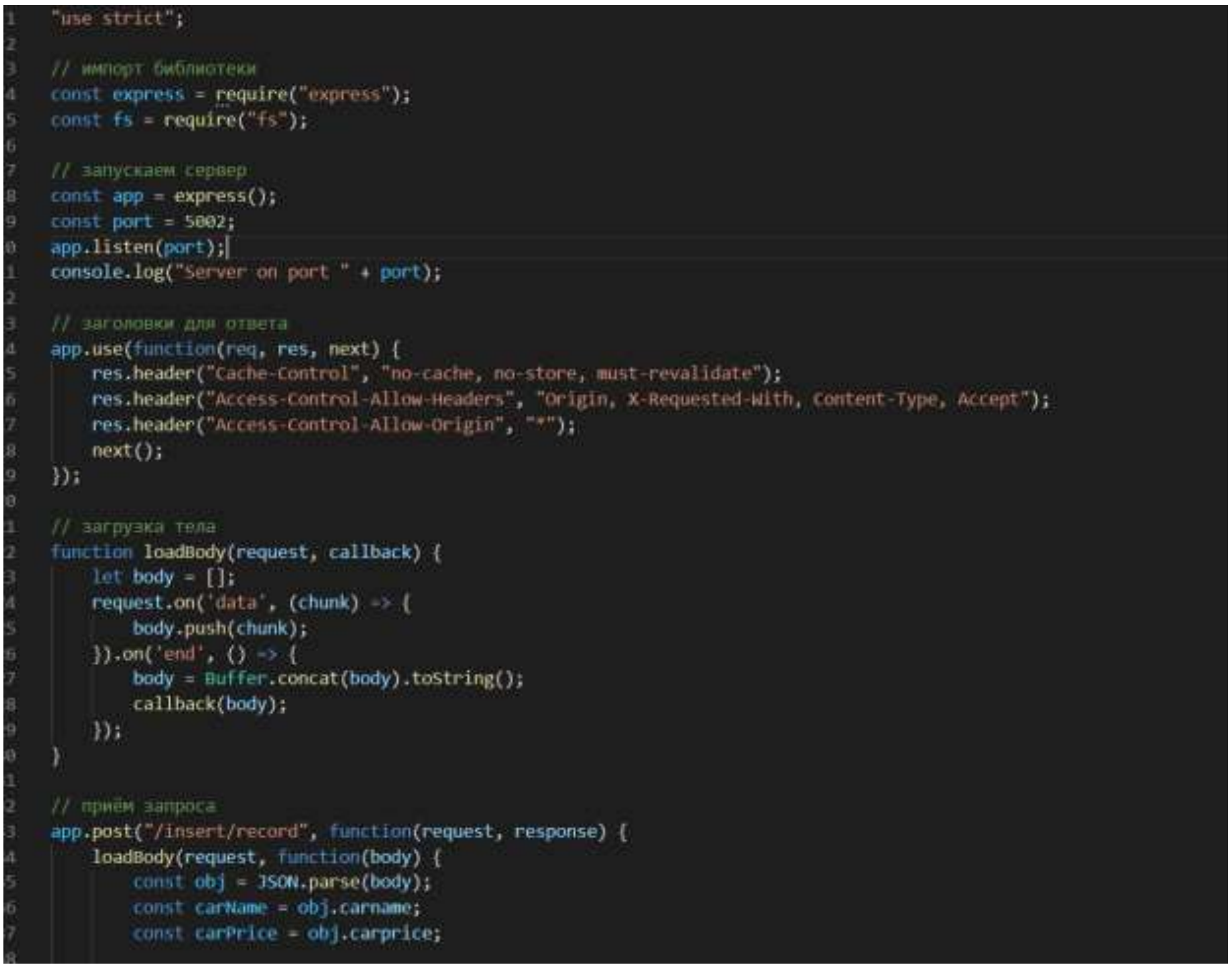
Группа ИУ7-55Б

Оценка (баллы) ________

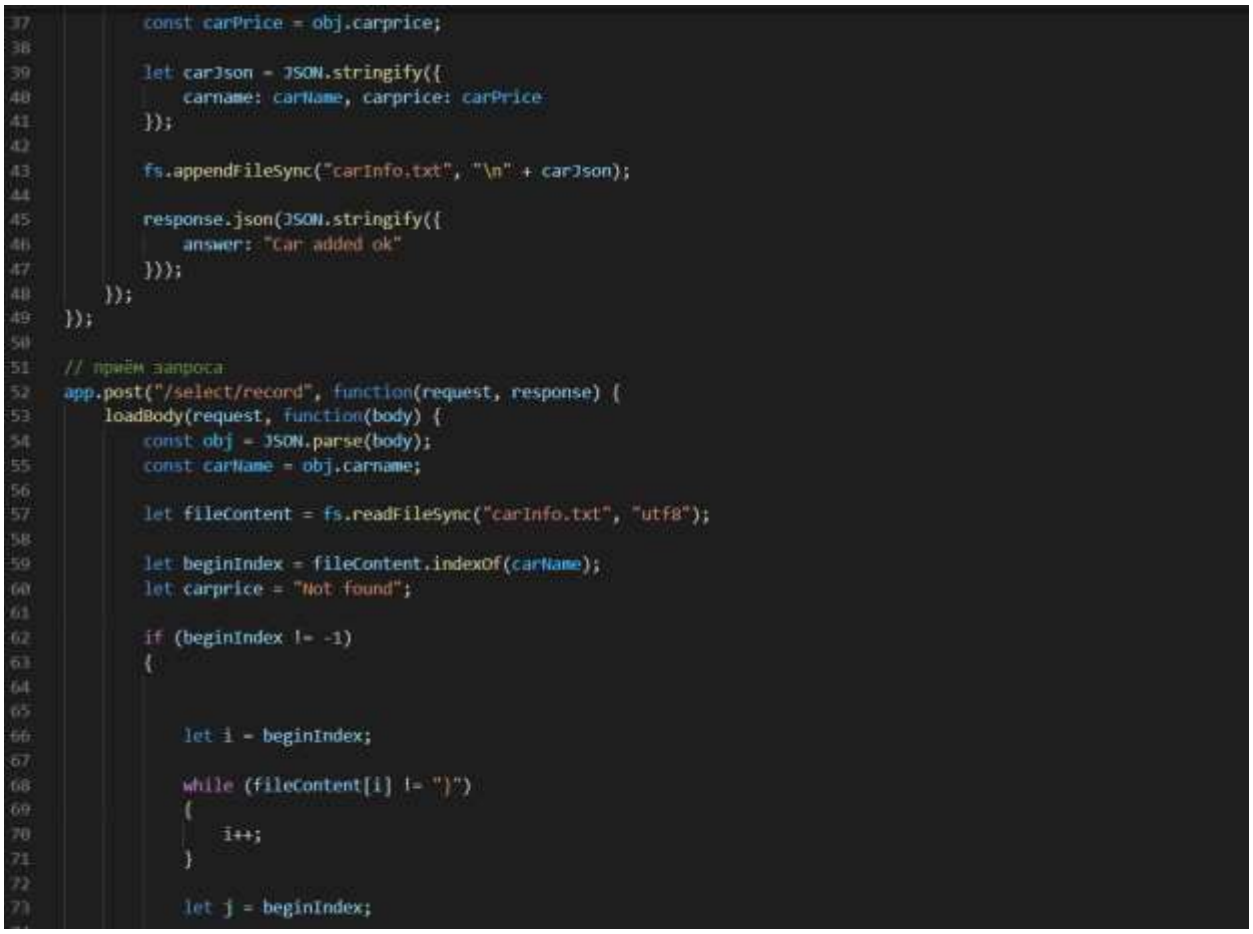
Москва.
2020 г.

Цель: Ознакомиться с созданием, запуском и взаимодействия нескольких серверов.

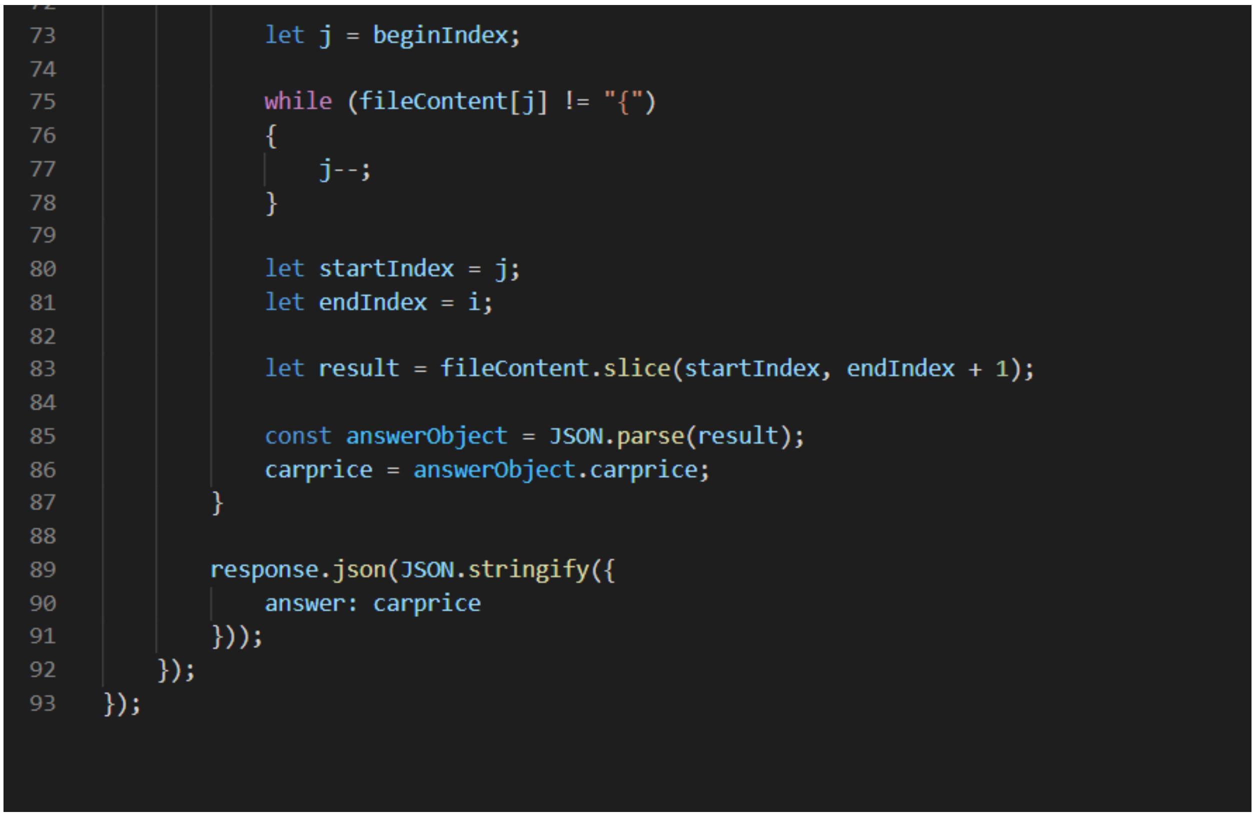
Task_7



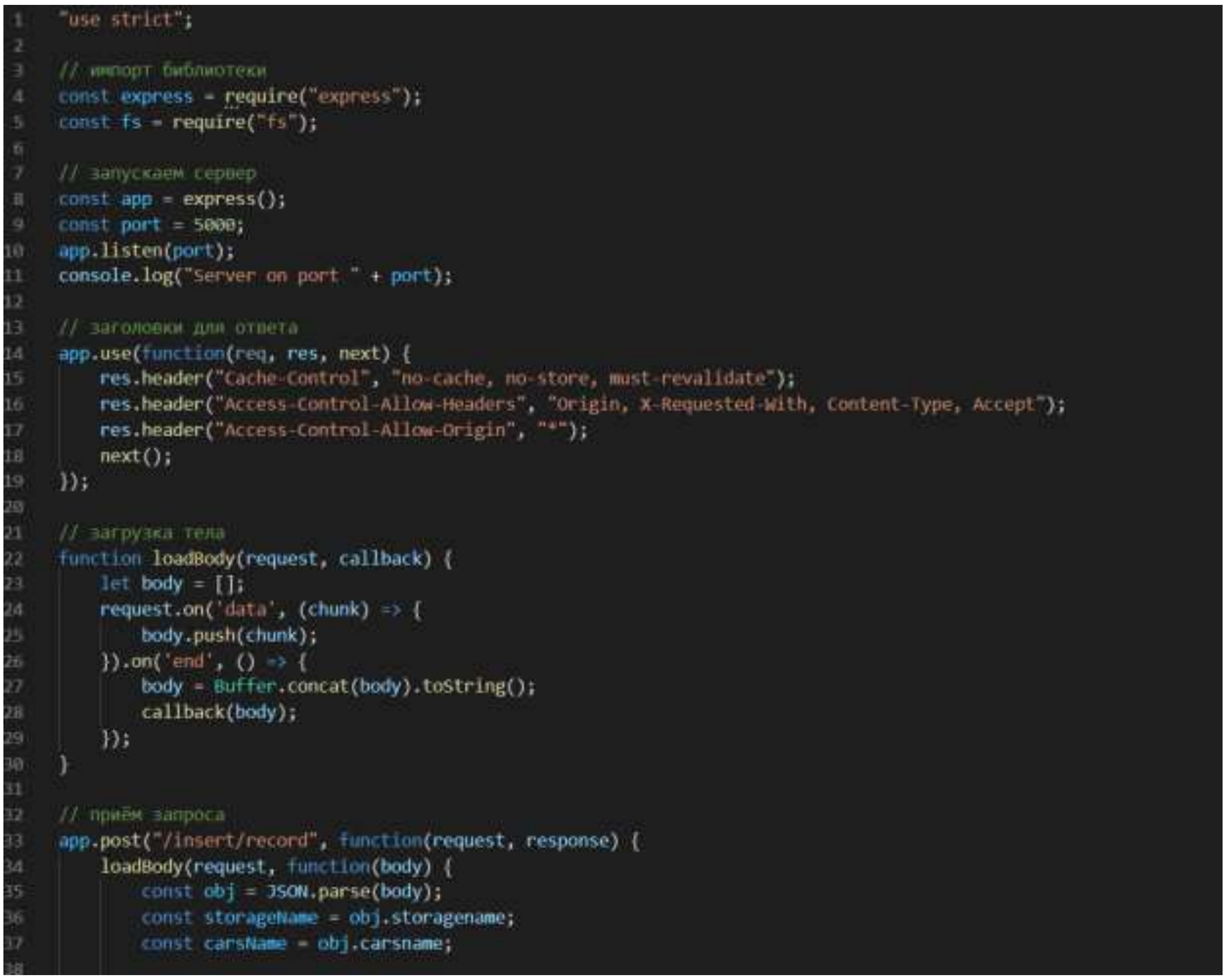
```
"use strict";

// импорт библиотеки
const express = require("express");
const fs = require("fs");

// запускаем сервер
const app = express();
const port = 5002;
app.listen(port);
console.log("Server on port " + port);

// заголовки для ответа
app.use(function(req, res, next) {
    res.header("Cache-Control", "no-cache, no-store, must-revalidate");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    res.header("Access-Control-Allow-Origin", "*");
    next();
});

// загрузка тела
function loadBody(request, callback) {
    let body = [];
    request.on('data', (chunk) => {
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        callback(body);
    });
}

// приём запроса
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const carName = obj.carname;
        const carPrice = obj.carprice;
```



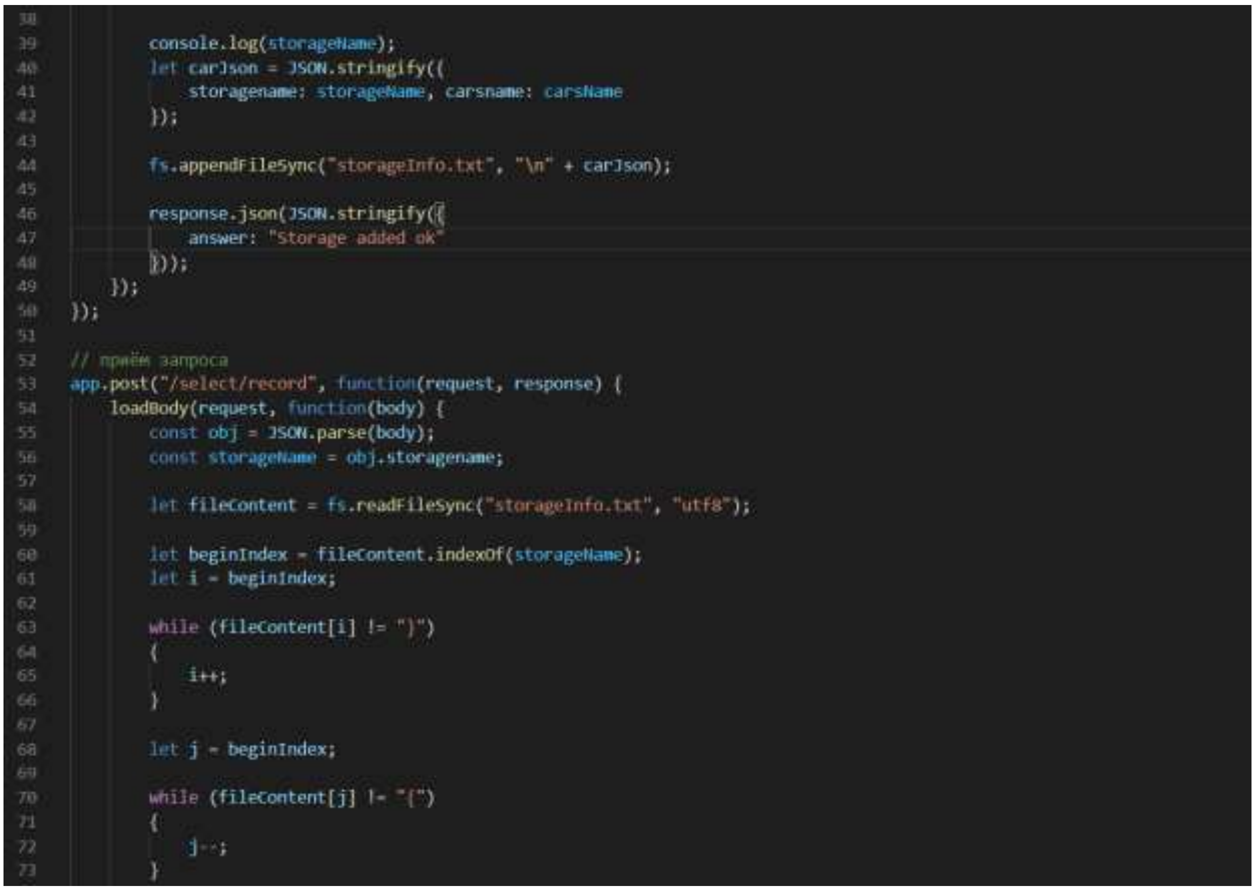
```
        const carPrice = obj.carprice;

        let carJson = JSON.stringify({
            carname: carName, carprice: carPrice
        });

        fs.appendFileSync("carInfo.txt", "\n" + carJson);

        response.json(JSON.stringify({
            answer: "Car added ok"
        }));
    });
});

// приём запроса
app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const carName = obj.carname;

        let fileContent = fs.readFileSync("carInfo.txt", "utf8");

        let beginIndex = fileContent.indexOf(carName);
        let carprice = "Not found";

        if (beginIndex != -1)
        {


            let i = beginIndex;

            while (fileContent[i] != "}")
            {
                i++;
            }

            let j = beginIndex;
```



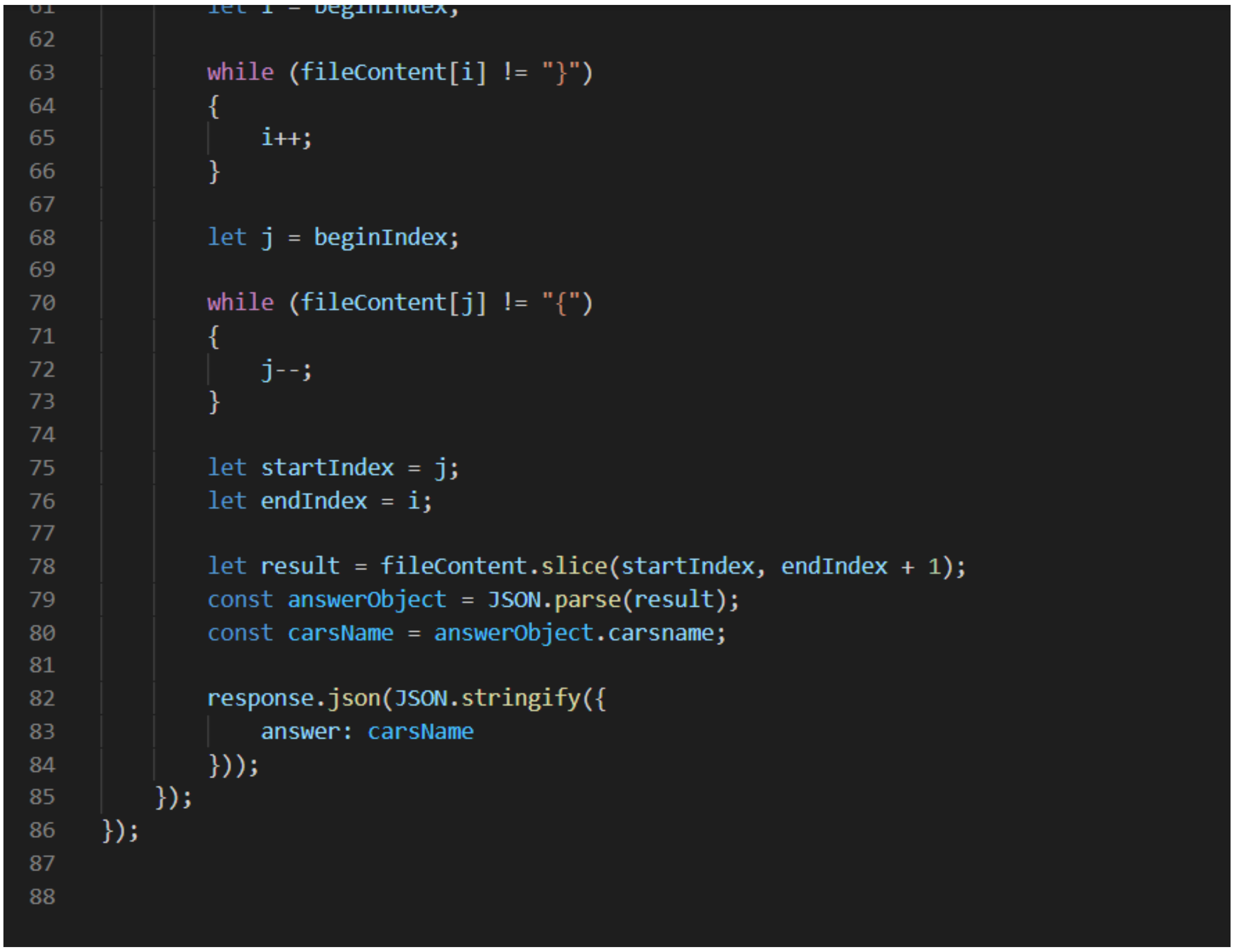
```
            let j = beginIndex;

            while (fileContent[j] != "{")
            {
                j--;
            }

            let startIndex = j;
            let endIndex = i;

            let result = fileContent.slice(startIndex, endIndex + 1);

            const answerObject = JSON.parse(result);
            carprice = answerObject.carprice;
        }

        response.json(JSON.stringify({
            answer: carprice
        }));
    });
});
```



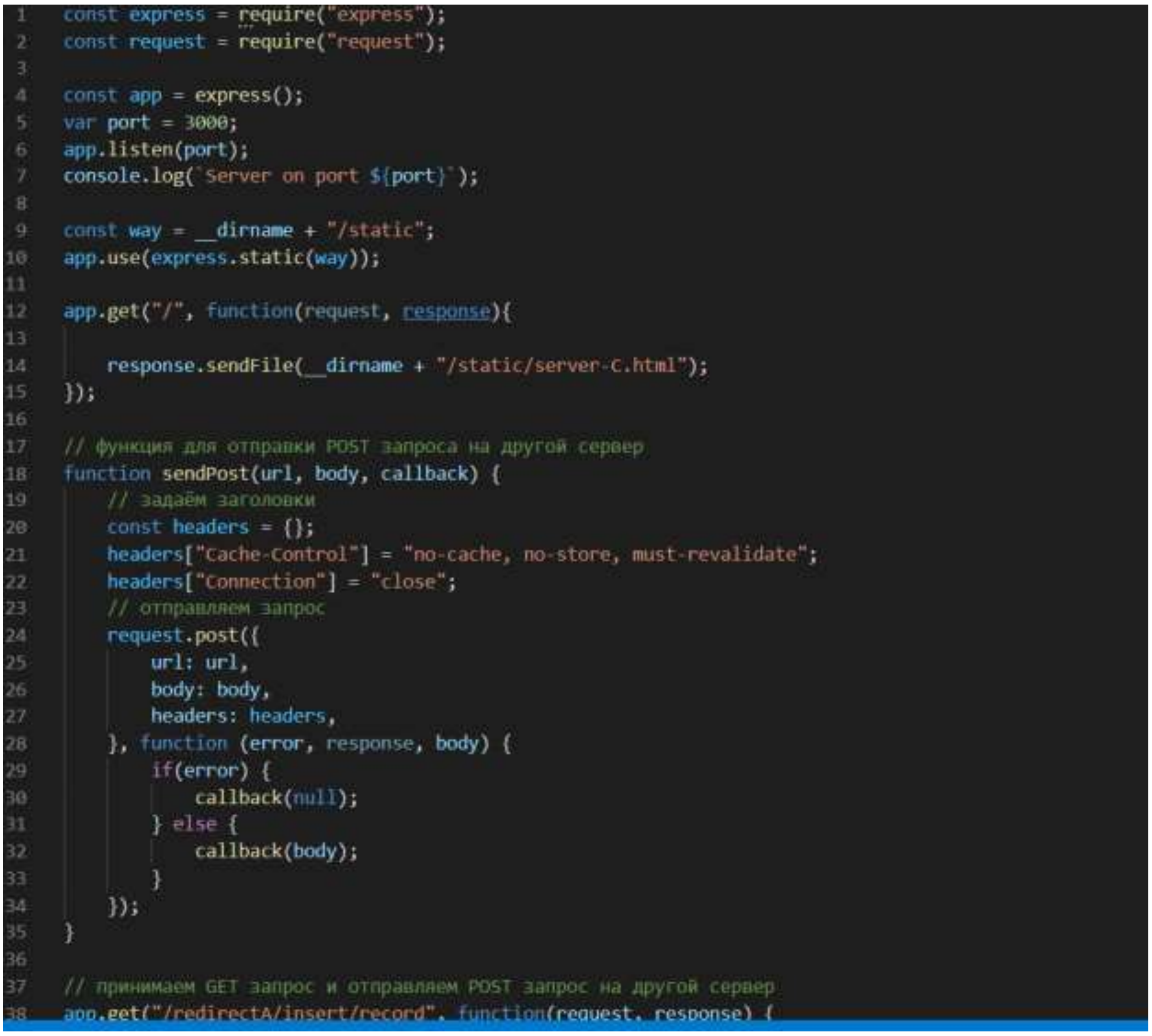
```javascript
"use strict";

// импорт библиотеки
const express = require("express");
const fs = require("fs");

// запускаем сервер
const app = express();
const port = 5000;
app.listen(port);
console.log("Server on port " + port);

// заголовки для ответа
app.use(function(req, res, next) {
    res.header("Cache-Control", "no-cache, no-store, must-revalidate");
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
    res.header("Access-Control-Allow-Origin", "*");
    next();
});

// загрузка тела
function loadBody(request, callback) {
    let body = [];
    request.on('data', (chunk) => {
        body.push(chunk);
    }).on('end', () => {
        body = Buffer.concat(body).toString();
        callback(body);
    });
}

// приём запроса
app.post("/insert/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const storageName = obj.storagename;
        const carsName = obj.carsname;

```



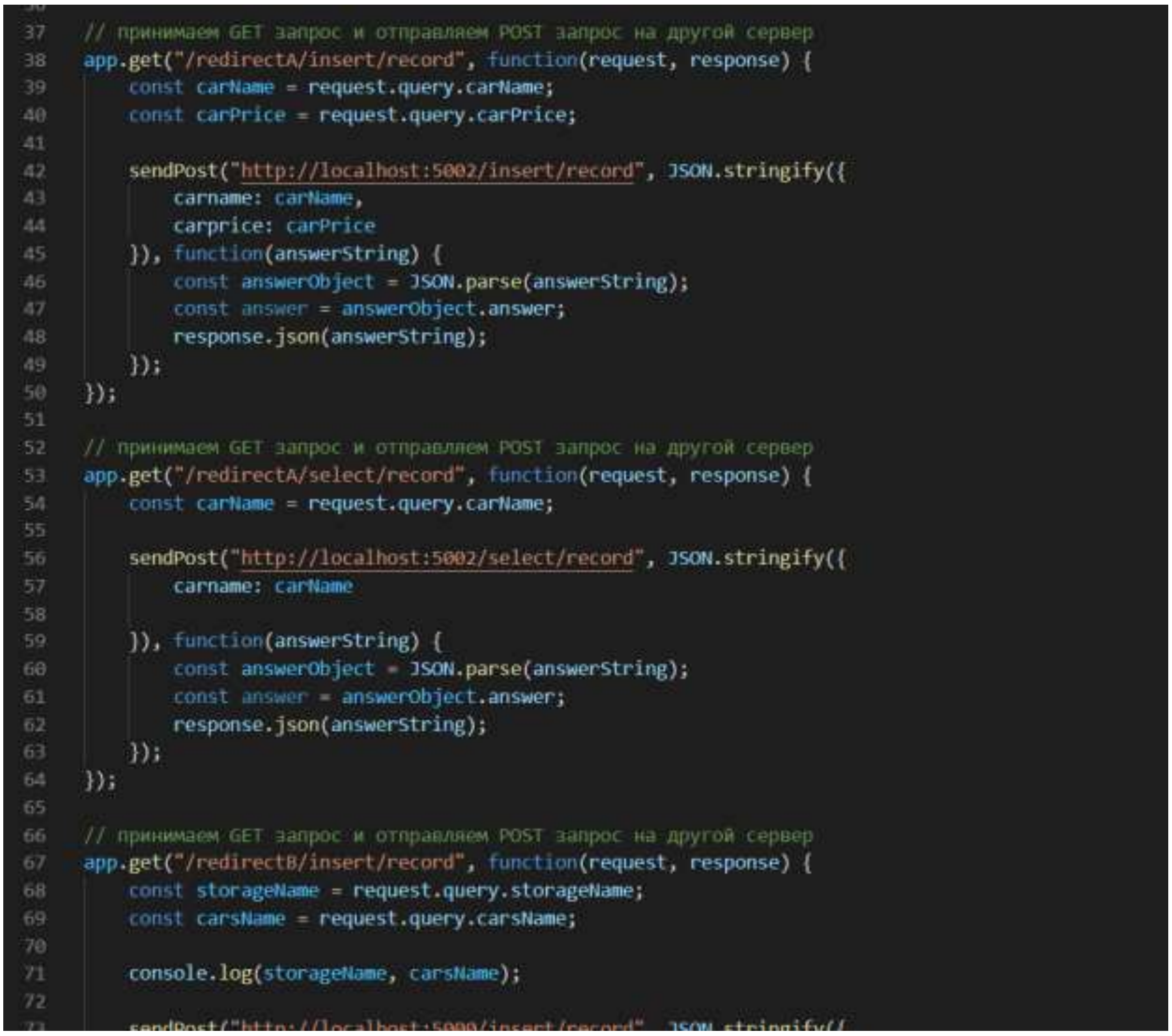
```javascript

        console.log(storageName);
        let carJson = JSON.stringify({
            storagename: storageName, carsname: carsName
        });

        fs.appendFileSync("storageInfo.txt", "\n" + carJson);

        response.json(JSON.stringify({
            answer: "Storage added ok"
        }));
    });
});

// приём запроса
app.post("/select/record", function(request, response) {
    loadBody(request, function(body) {
        const obj = JSON.parse(body);
        const storageName = obj.storagename;

        let fileContent = fs.readFileSync("storageInfo.txt", "utf8");

        let beginIndex = fileContent.indexOf(storageName);
        let i = beginIndex;

        while (fileContent[i] != "}")
        {
            i++;
        }

        let j = beginIndex;

        while (fileContent[j] != "{")
        {
            j--;
        }
```



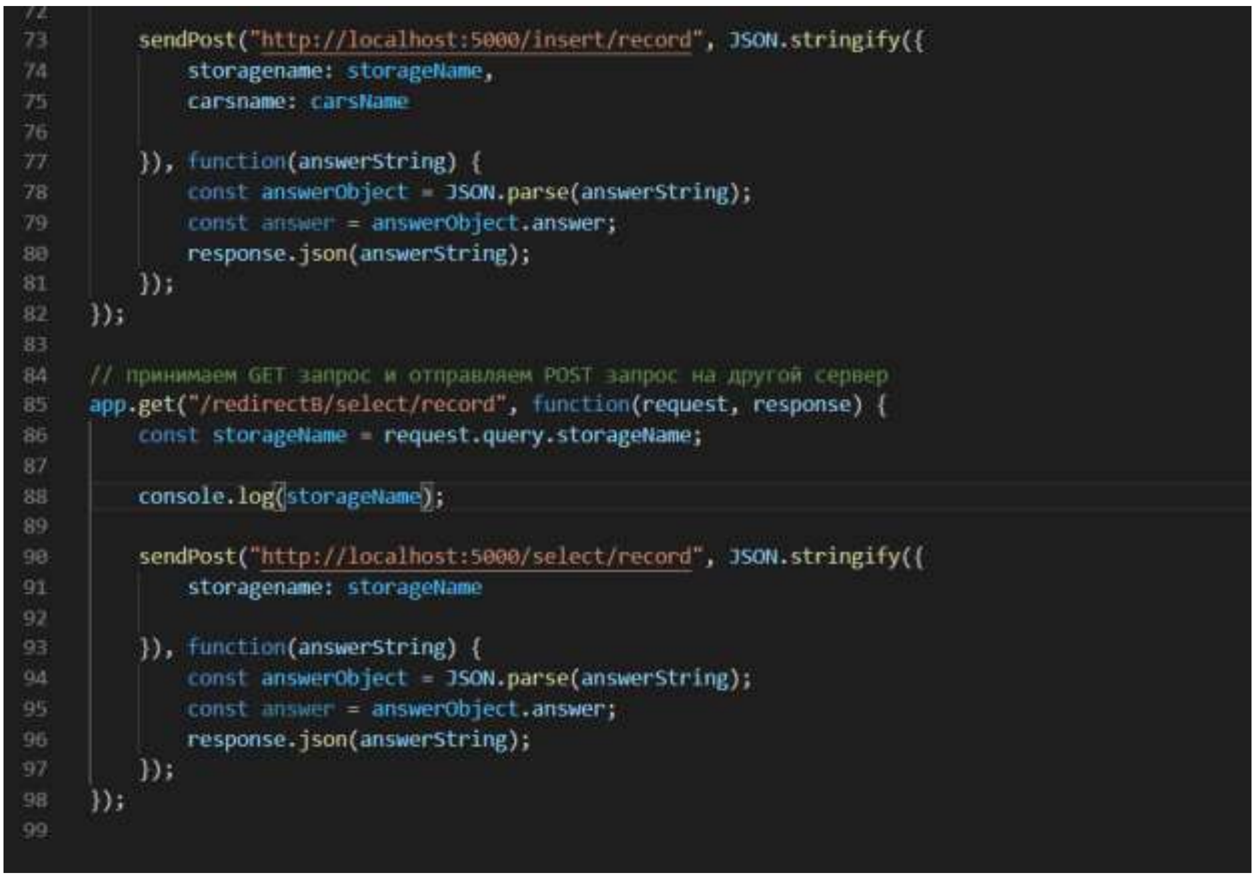
```
        let i = beginIndex;

        while (fileContent[i] != "}")
        {
            i++;
        }

        let j = beginIndex;

        while (fileContent[j] != "{")
        {
            j--;
        }

        let startIndex = j;
        let endIndex = i;

        let result = fileContent.slice(startIndex, endIndex + 1);
        const answerObject = JSON.parse(result);
        const carsName = answerObject.carsname;

        response.json(JSON.stringify({
            answer: carsName
        }));
    });
});
```



```
const express = require("express");
const request = require("request");

const app = express();
var port = 3000;
app.listen(port);
console.log(`Server on port ${port}`);

const way = __dirname + "/static";
app.use(express.static(way));

app.get("/", function(request, response){

    response.sendFile(__dirname + "/static/server-C.html");
});

// функция для отправки POST запроса на другой сервер
function sendPost(url, body, callback) {
    // задаём заголовки
    const headers = {};
    headers["Cache-Control"] = "no-cache, no-store, must-revalidate";
    headers["Connection"] = "close";
    // отправляем запрос
    request.post({
        url: url,
        body: body,
        headers: headers,
    }, function (error, response, body) {
        if(error) {
            callback(null);
        } else {
            callback(body);
        }
    });
}

// принимаем GET запрос и отправляем POST запрос на другой сервер
app.get("/redirectA/insert/record", function(request, response) {
```

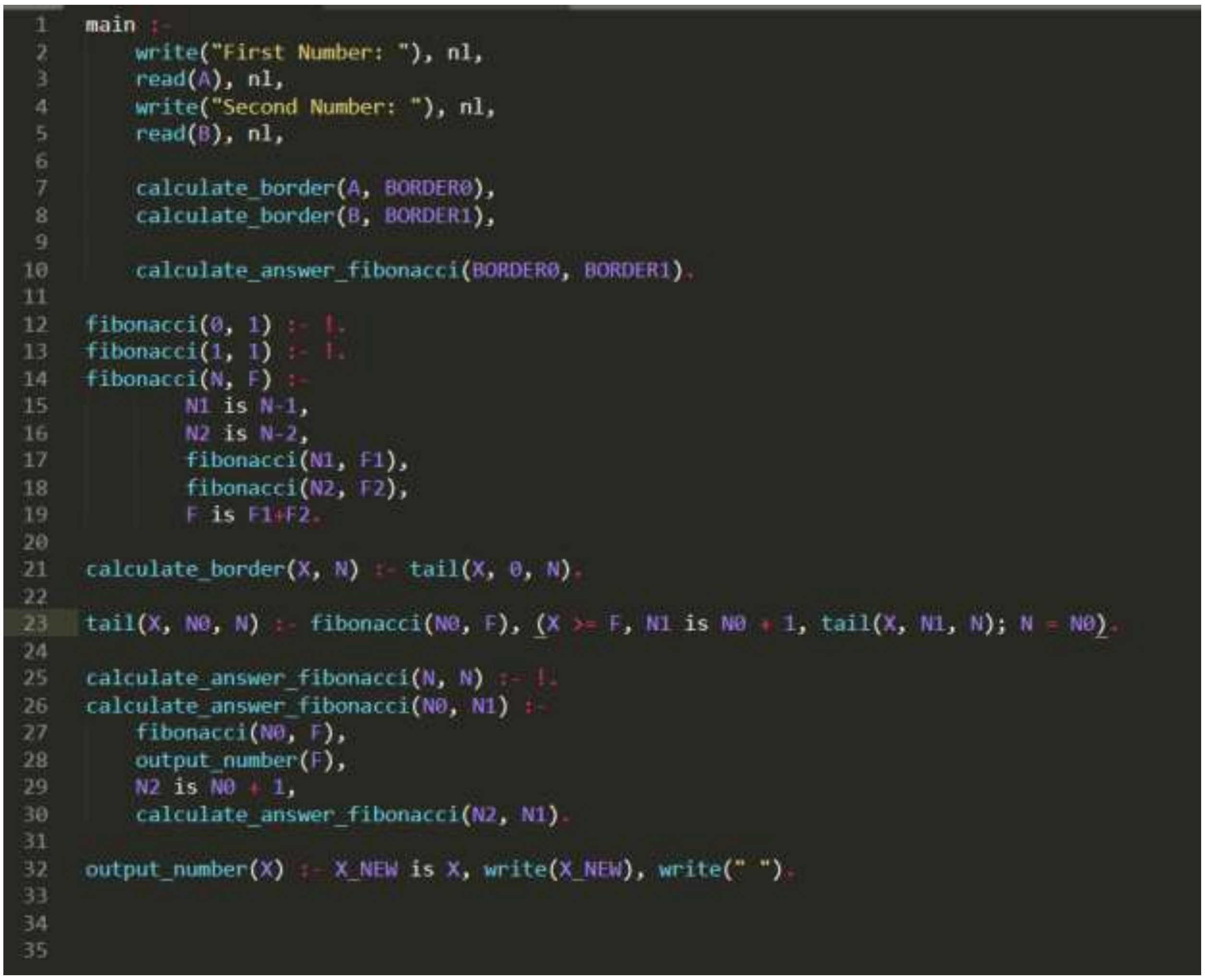


```
// принимаем GET запрос и отправляем POST запрос на другой сервер
app.get("/redirectA/insert/record", function(request, response) {
    const carName = request.query.carName;
    const carPrice = request.query.carPrice;

    sendPost("http://localhost:5002/insert/record", JSON.stringify({
        carname: carName,
        carprice: carPrice
    }), function(answerString) {
        const answerObject = JSON.parse(answerString);
        const answer = answerObject.answer;
        response.json(answerString);
    });
});

// принимаем GET запрос и отправляем POST запрос на другой сервер
app.get("/redirectA/select/record", function(request, response) {
    const carName = request.query.carName;

    sendPost("http://localhost:5002/select/record", JSON.stringify({
        carname: carName

    }), function(answerString) {
        const answerObject = JSON.parse(answerString);
        const answer = answerObject.answer;
        response.json(answerString);
    });
});

// принимаем GET запрос и отправляем POST запрос на другой сервер
app.get("/redirectB/insert/record", function(request, response) {
    const storageName = request.query.storageName;
    const carsName = request.query.carsName;

    console.log(storageName, carsName);

    sendPost("http://localhost:5000/insert/record", JSON.stringify({
```

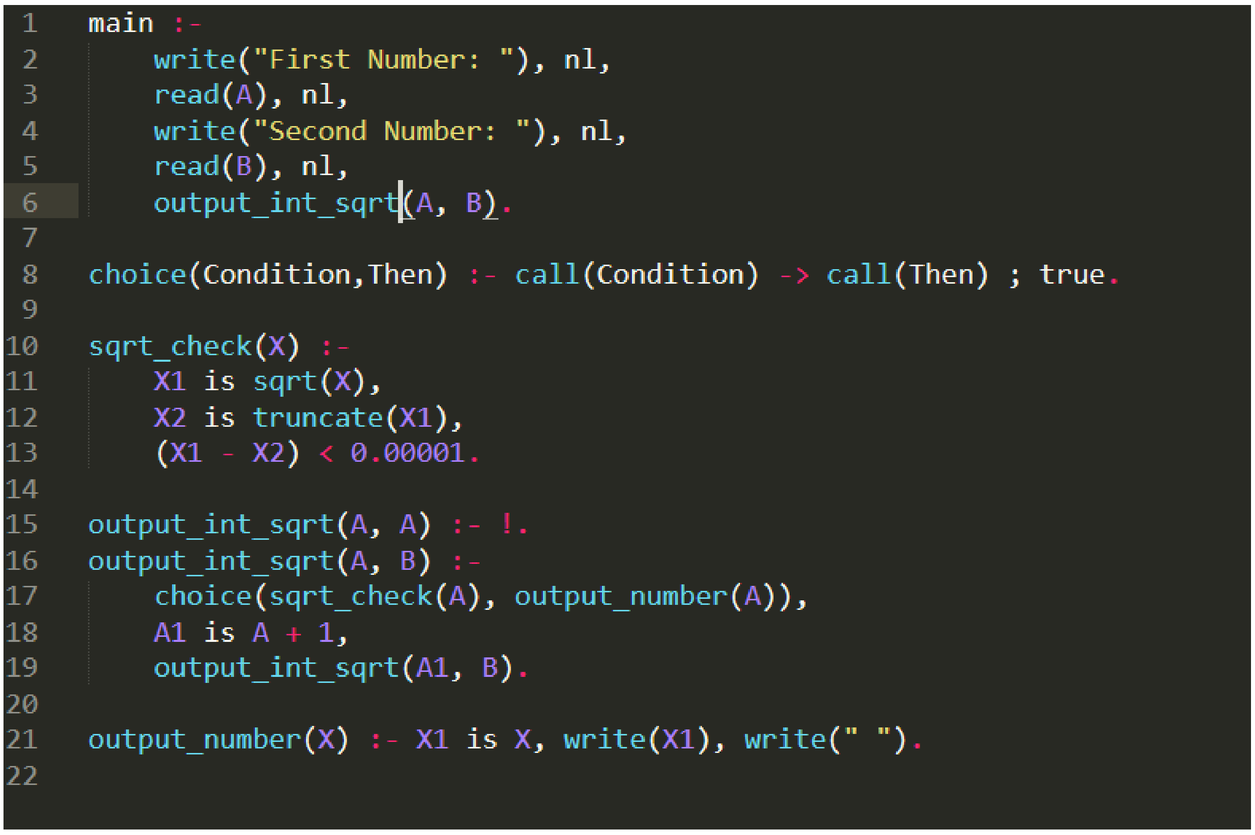


```
    sendPost("http://localhost:5000/insert/record", JSON.stringify({
        storagename: storageName,
        carsname: carsName

    }), function(answerString) {
        const answerObject = JSON.parse(answerString);
        const answer = answerObject.answer;
        response.json(answerString);
    });
});

// принимаем GET запрос и отправляем POST запрос на другой сервер
app.get("/redirectB/select/record", function(request, response) {
    const storageName = request.query.storageName;

    console.log(storageName);

    sendPost("http://localhost:5000/select/record", JSON.stringify({
        storagename: storageName

    }), function(answerString) {
        const answerObject = JSON.parse(answerString);
        const answer = answerObject.answer;
        response.json(answerString);
    });
});

```

Task_7



```prolog
main :-
    write("First Number: "), nl,
    read(A), nl,
    write("Second Number: "), nl,
    read(B), nl,

    calculate_border(A, BORDER0),
    calculate_border(B, BORDER1),

    calculate_answer_fibonacci(BORDER0, BORDER1).

fibonacci(0, 1) :- !.
fibonacci(1, 1) :- !.
fibonacci(N, F) :-
        N1 is N-1,
        N2 is N-2,
        fibonacci(N1, F1),
        fibonacci(N2, F2),
        F is F1+F2.

calculate_border(X, N) :- tail(X, 0, N).

tail(X, N0, N) :- fibonacci(N0, F), (X >= F, N1 is N0 + 1, tail(X, N1, N); N = N0).

calculate_answer_fibonacci(N, N) :- !.
calculate_answer_fibonacci(N0, N1) :-
    fibonacci(N0, F),
    output_number(F),
    N2 is N0 + 1,
    calculate_answer_fibonacci(N2, N1).

output_number(X) :- X_NEW is X, write(X_NEW), write(" ").
```



```prolog
main :-
    write("First Number: "), nl,
    read(A), nl,
    write("Second Number: "), nl,
    read(B), nl,
    output_int_sqrt(A, B).

choice(Condition,Then) :- call(Condition) -> call(Then) ; true.

sqrt_check(X) :-
    X1 is sqrt(X),
    X2 is truncate(X1),
    (X1 - X2) < 0.00001.

output_int_sqrt(A, A) :- !.
output_int_sqrt(A, B) :-
    choice(sqrt_check(A), output_number(A)),
    A1 is A + 1,
    output_int_sqrt(A1, B).

output_number(X) :- X1 is X, write(X1), write(" ").
```

Вывод: Я ознакомился с созданием, запуском и взаимодействия нескольких серверов.