Part 1:
Question 1:Higher order function:

```
function processPost(posts){
    console.log("Exctracting title and body of each post:");
    for(let i=0;i<posts.length;i++){
        console.log("Title:"+ posts[i].title);
        console.log("Body:" +posts[i].body);
    }
    let titlesAndInt=posts.map(function(p){
        return {id:p.id,title:p.title};
    });
    console.log("Titles and ids:",titlesAndInt);

    let user1Posts=post.filter(function(P){
        return P.user1Posts;
    })
    console.log("post from userid:",user1Posts);


    let postCountByUser=posts.reduce(function(acc,p){
        if(acc[p.userId]){
            acc[p.userId]=acc[p.userId]+1;

        }
        else{
            acc[p.userId]=1;
        }
        return acc;
    },
    {});
    console.log("Post count by user:");
    console.log(postCountByUser);

    let longTitle=posts.some(function(p){
        return p.title.length>50;
    });
    console.log("And title longer than 50 chars?");
    console.log(longTitle);

    let sortedPosts=post.sort(function(a,b){
        if(a.title<b.title){
            return -1;
        }
        else if(a.title>b.title){
```

```
            return 1;
        }
        else{
            return 0;
        }
    });
    console.log("Posts sorted by title:");
    console.log(sortedPosts);
    let user7Titles =posts.filter(function(p){
        return p.userId===7;
    }).map(function(p){
        return p.title;
    });
    console.log("titles of posts from userId 7:");
    console.log(user7Titles);

}

fetch("https://jsonplaceholder.typicode.com/posts").then(function(res){
    return res.json();
})
.then(function(data){
    console.log("fetched posts succesfully");
    processPost(data);
})
.catch(function(err){
    console.log("error:"+err);
})
```

Question 2: async function:

```
import{processPost} from 'C:\Users\Smart\OneDrive\ドキュメント
\Desktop\Evaluation-06-09-25\Part1_HigherOderFunction.js'
const fetchAndProcessPosts=async()=>{
    try{
        const response=await fetch("https://jsonplaceholder.typicode.com/posts");
        const data=await response.json();
        console.log("fetched posts succesfully!");
        processPost(data);

    }
    catch(error){
        console.log("Error fetching posts:",error);
```

```
    }
};
```

—----------

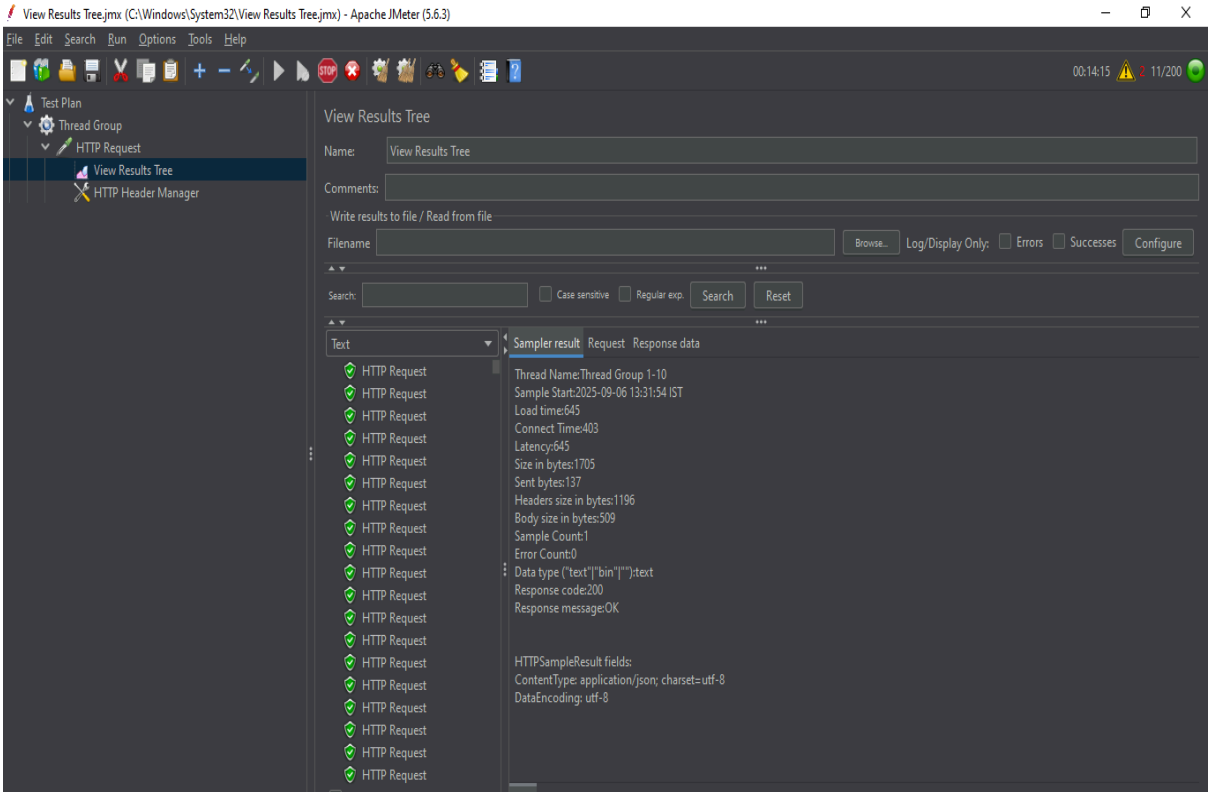
Question 3 : Demonstrate feature of ES6

- Const: user for variable that not change
- Arrow function show function information
- Template literal use for embedding variables directly

—----------

Part 2: Performance Testing :

Objective : test the performance of endpoint : /api/user/profile/{id}
Scope: endepoint /api/user/profile/{id}
Test user: 200 concurrent user
Test duration:60 sec
Scenario: 200 concurrent user visit the endpoint continuous for 60sec
Steps: open jmeter add thread group then add request
Path: api/user/profile/{id}
Number of threads:200
Ramp-up-time:1sec
Loop time : 60sec
Response time(load time):645ms
Throughput time: 3 request per second
Error role: 0


View Results Tree.jmx (C:\Windows\System32\View Results Tree.jmx) - Apache JMeter (5.6.3)
File Edit Search Run Options Tools Help
00:14:15 2 11/200
Test Plan
Thread Group
HTTP Request
View Results Tree
HTTP Header Manager
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Search:
Case sensitive
Regular exp.
Search
Reset
Text
HTTP Request
Sampler result
Request
Response data
Thread Name:Thread Group 1-10
Sample Start:2025-09-06 13:31:54 IST
Load time:645
Connect Time:403
Latency:645
Size in bytes:1705
Sent bytes:137
Headers size in bytes:1196
Body size in bytes:509
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:OK
HTTPSampleResult fields:
ContentType: application/json; charset=utf-8
DataEncoding: utf-8