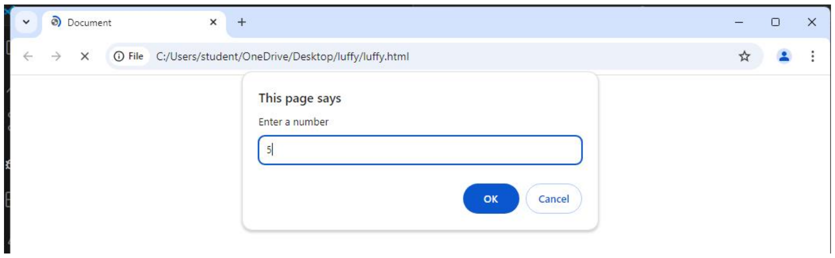
1. Recursion and stack:

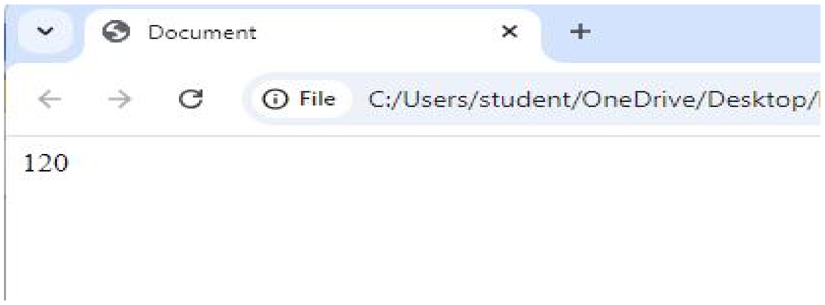
Task 1: Implement a function to calculate the factorial of a number using recursion.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var n= Number(prompt("Enter a number"));
        function fact(n)
        {
            if(n==0)
            return 1;
        else
        return n*fact(n-1);
        }
        document.writeln(fact(n));
    </script>
</body>
</html>
```

Output:



```
This page says
Enter a number
5
```

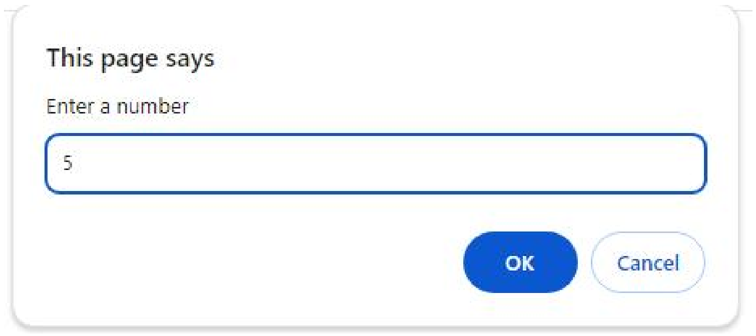


```
120
```

Task 2: Write a recursive function to find the nth Fibonacci number.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var n= Number(prompt("Enter a number"));
        function fib(n)
        {
            if(n<=1)
            return n;
        else
        return fib(n-1)+fib(n-2);
        }
        document.writeln(fib(n-1));
    </script>
</body>
</html>
```

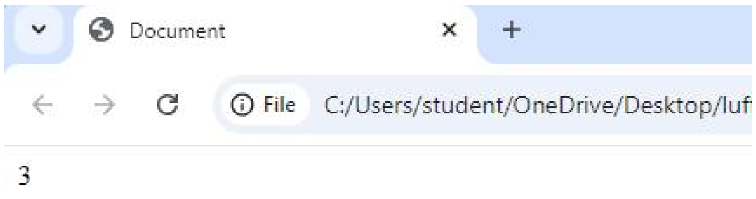
Output:



This page says

Enter a number

```
5
```



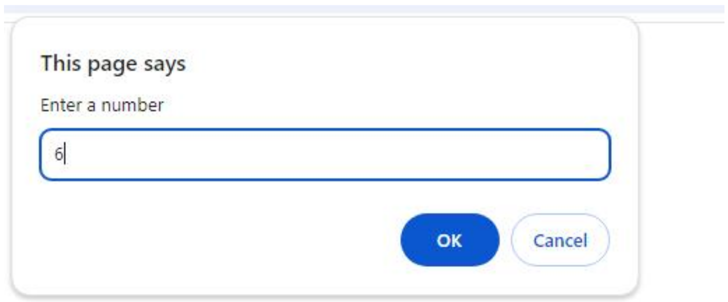
```
3
```

Task 3: Create a function to determine the total number of ways one can climb a

staircase with 1, 2, or 3 steps at a time using recursion.

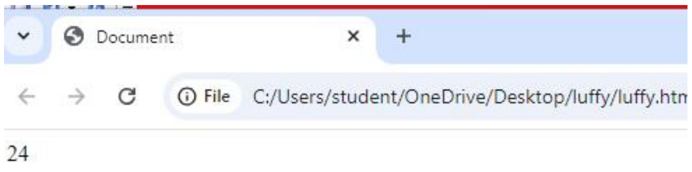
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var n= Number(prompt("Enter a number"));
        function step(n)
        {
            if(n==0)
            return 1;
            if(n<0)
            return 0;
        return step(n-1)+step(n-2)+step(n-3);
        }
        document.writeln(step(n));
    </script>
</body>
</html>
```

Output:


This page says

Enter a number

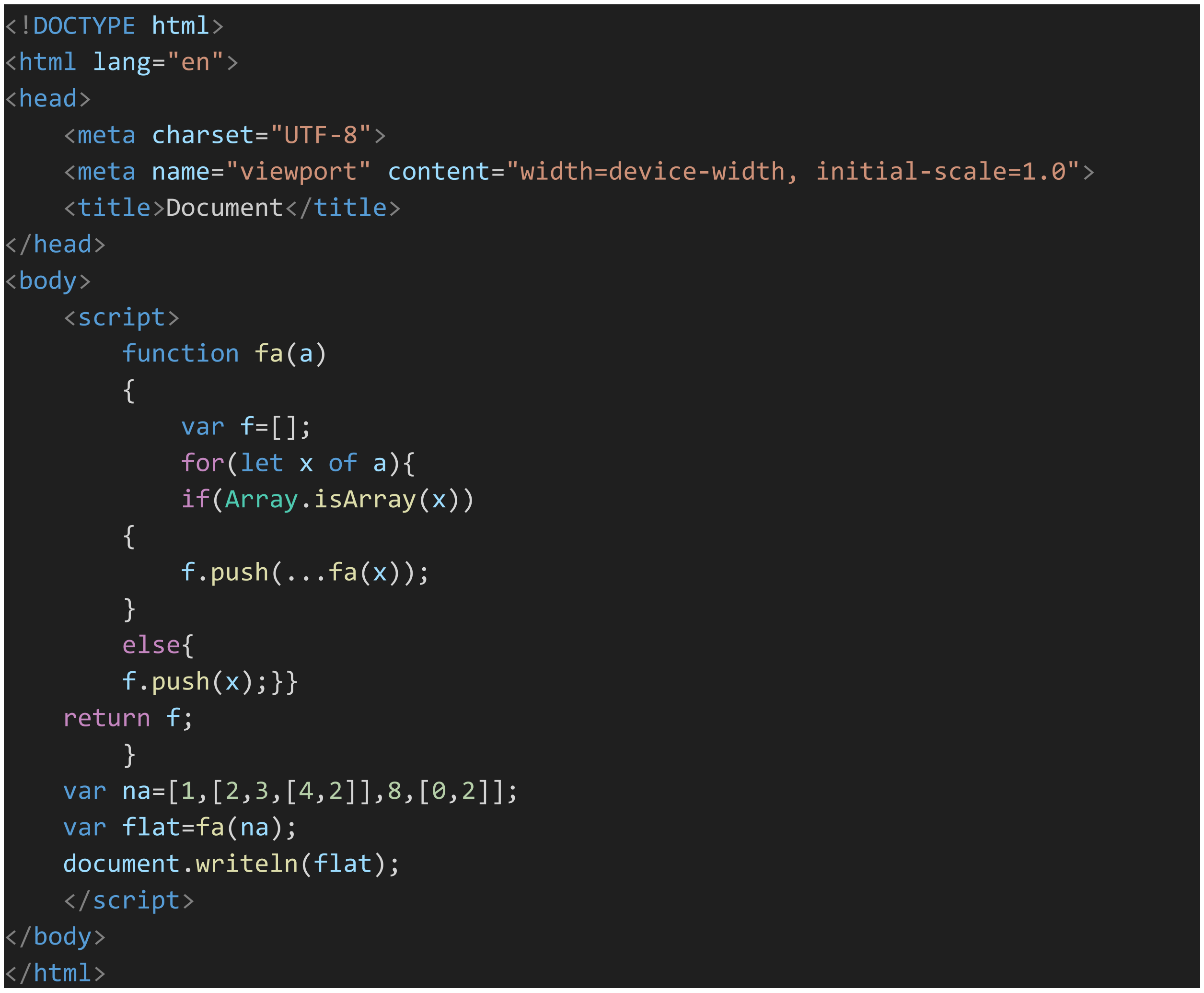
6



```
C:/Users/student/OneDrive/Desktop/luffy/luffy.htm
```

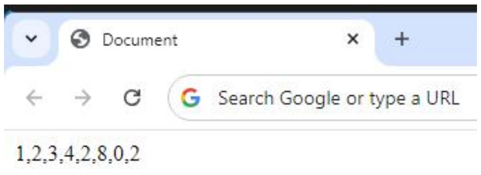
```
24
```

Task 4: Write a recursive function to flatten a nested array structure.



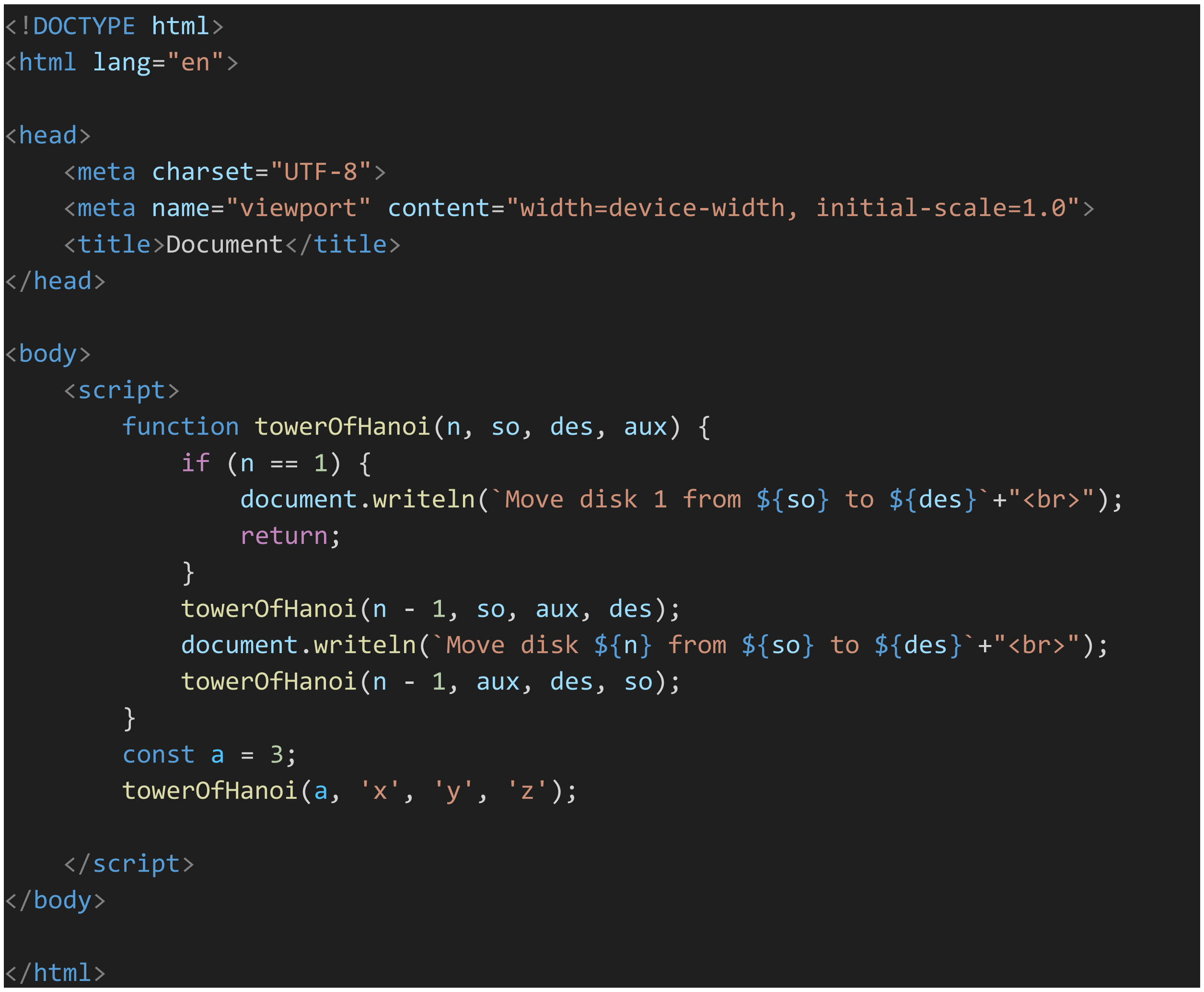
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function fa(a)
        {
            var f=[];
            for(let x of a){
            if(Array.isArray(x))
        {
            f.push(...fa(x));
        }
        else{
        f.push(x);}}
    return f;
        }
    var na=[1,[2,3,[4,2]],8,[0,2]];
    var flat=fa(na);
    document.writeln(flat);
    </script>
</body>
</html>
```

Output:



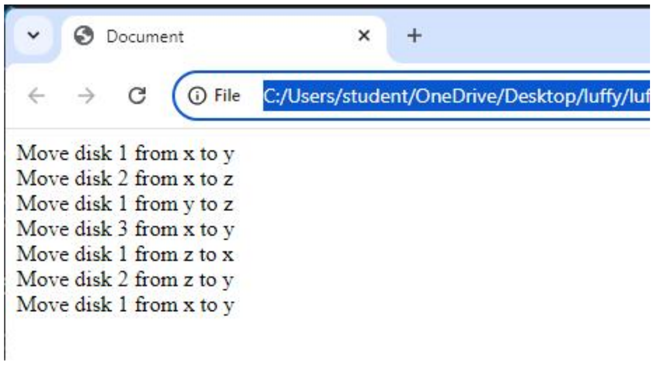
```
1,2,3,4,2,8,0,2
```

Task 5: Implement the recursive Tower of Hanoi solution.



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>
        function towerOfHanoi(n, so, des, aux) {
            if (n == 1) {
                document.writeln(`Move disk 1 from ${so} to ${des}`+"<br>");
                return;
            }
            towerOfHanoi(n - 1, so, aux, des);
            document.writeln(`Move disk ${n} from ${so} to ${des}`+"<br>");
            towerOfHanoi(n - 1, aux, des, so);
        }
        const a = 3;
        towerOfHanoi(a, 'x', 'y', 'z');

    </script>
</body>

</html>
```

Output:



```
Move disk 1 from x to y
Move disk 2 from x to z
Move disk 1 from y to z
Move disk 3 from x to y
Move disk 1 from z to x
Move disk 2 from z to y
Move disk 1 from x to y
```

2. JSON and variable length arguments/spread syntax:

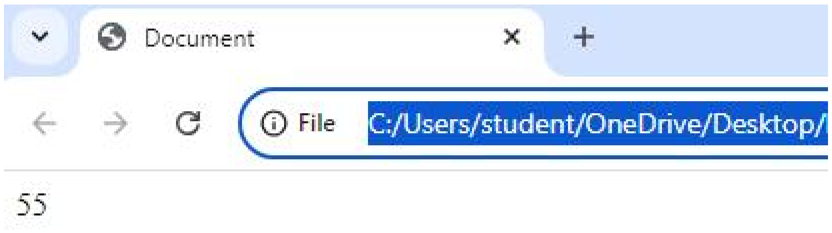
Task 1: Write a function that takes an arbitrary number of arguments and returns their

sum.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>
       function arb(...a)
       {
        var t=0;
        for(let m of a)
       {
        t+=m;
       }
       return t;
    }
```

```
       var x=arb(1,2,3,4,5,6,7,8,9,10)
       document.writeln(x)

    </script>
</body>

</html>
```

Output:


Document

File C:/Users/student/OneDrive/Desktop/

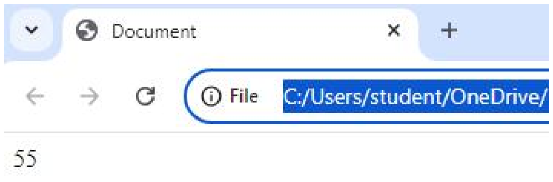
```
55
```

Task 2: Modify a function to accept an array of numbers and return their sum using the

spread syntax.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>
       function s(...a)
       {
        var t=0;
        for(let m of a)
       {
        t+=m;
       }
       return t;
    }
    function arb(a)
    {
        return s(...a);
    }

       var x=arb([1,2,3,4,5,6,7,8,9,10])
       document.writeln(x)

    </script>
</body>

</html>
```

Output:



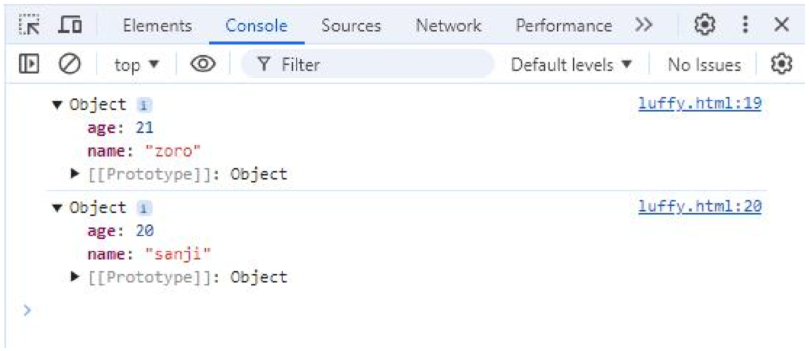
```
55
```

Task 3: Create a deep clone of an object using JSON methods.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>

      var obj1={
        name:'zoro',age:21
      };
      var obj2 =JSON.parse(JSON.stringify(obj1));
      obj2.name='sanji';
      obj2.age=20;
      console.log(obj1);
      console.log(obj2);
```

```
</script>
</body>

</html>
```

Output:



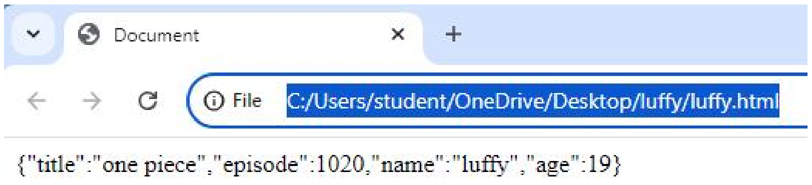
```
▼ Object i                                          luffy.html:19
    age: 21
    name: "zoro"
  ▶ [[Prototype]]: Object
▼ Object i                                          luffy.html:20
    age: 20
    name: "sanji"
  ▶ [[Prototype]]: Object
>
```

Task 4: Write a function that returns a new object, merging two provided objects using

the spread syntax.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>

      var obj1={
        title:'one piece',
        episode:1020
      };
      var obj2={
        name:'luffy',age:19
      };
      var obj={
      ...obj1,...obj2
      };

      document.writeln(JSON.stringify(obj));
      console.log(obj);

</script>
</body>

</html>
```

Output:


Document

File C:/Users/student/OneDrive/Desktop/luffy/luffy.html

```
{"title":"one piece","episode":1020,"name":"luffy","age":19}
```

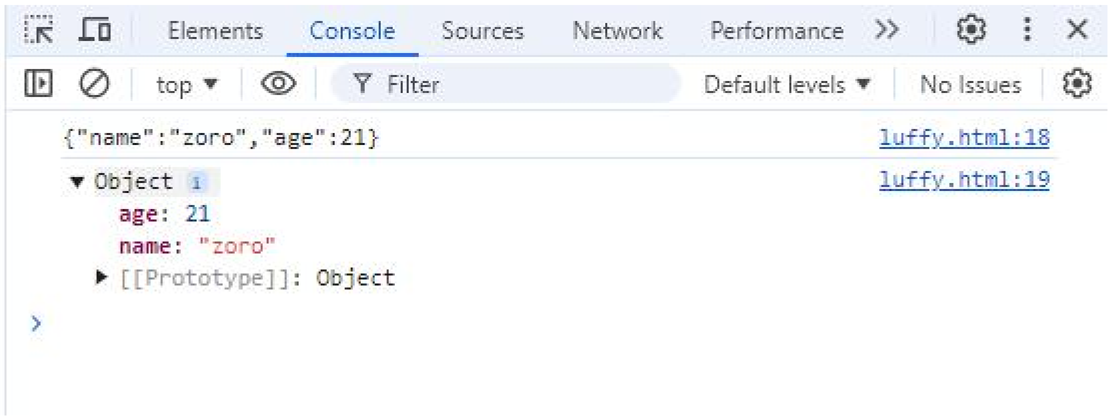
Task 5: Serialize a JavaScript object into a JSON string and then parse it back into an

object.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>

      var obj1={
        name:'zoro',age:21
      };
      var obj =JSON.stringify(obj1);
      var ob=JSON.parse(obj);
      console.log(obj);
      console.log(ob);
```

```
</script>
</body>

</html>
```

Output:



```
{"name":"zoro","age":21}
▼ Object
    age: 21
    name: "zoro"
  ▶ [[Prototype]]: Object
```