ES6 Assignments

1. **Symbols:** Write a class that defines next() method to return next number from Fibonacci series. The class will have a private attributes previousNo' & 'currentNo'.

```
const _previousNo = Symbol('previousNo');
const _currentNo = Symbol('currentNo');
class FibonacciSeries {
  constructor(previousNo, currentNo) {
    this[_previousNo] = previousNo;
    this[_currentNo] = currentNo;
  }
  next(num) {
    let _nextTerm = 0;
    for (let i = 0; i < num; i++) {
      _nextTerm = this[_previousNo] + this[_currentNo];
      this[_previousNo] = this[_currentNo];
      this[_currentNo] = _nextTerm;
    }
    return this[_currentNo];
  }
}
let fibo = new FibonacciSeries(0, 1);
console.log(fibo.next(4));
```

```
>> 5
```

2. **Iterators:** Write a program that returns next Armstrong number after calling getNextArmstrong() method.

```
function getArmstrongNumbers(start,end) {
  let armstrongArray = [];
  for (let i = start; i <= end; i++) {
      let numberOfDigits = i.toString().length;
      let sum = 0;
      let temp = i;
      while (temp > 0) {
          let remainder = temp % 10;
          sum += remainder ** numberOfDigits;
          temp = parseInt(temp / 10);
      }
      if (sum == i) {
          armstrongArray.push(i);
      }
  }
  return armstrongArray;
}

let getNextArmstrong = () => {
  let count = 0
  return {
      next: () => {
          let arr = getArmstrongNumbers(100,1000);
          return count < arr.length ?
              { armstrong_value: arr[count++], done: false } :
              { value: undefined, done: true }

      }
  }

}
const armstrongIterator = getNextArmstrong();
console.log(armstrongIterator.next());
console.log(armstrongIterator.next());
console.log(armstrongIterator.next());
console.log(armstrongIterator.next());
console.log(armstrongIterator.next());
```

```
>> {"armstrong_value":153,"done":false}
>> {"armstrong_value":370,"done":false}
>> {"armstrong_value":371,"done":false}
>> {"armstrong_value":407,"done":false}
>> {"value": undefined,"done": true}
```

3. **Generators:** Write a program that returns next Armstrong number after calling getNextArmstrong() method. Add functionality to reset generating Armstrong number from zero. In case, Armstrong number goes above one thousand then throw an error.

```javascript
function getArmstrongNumbers(start,end) {
  let armstrongArray = [];
  if (end>1000){
    return 'Maximum limit exceeded'
  }
  for (let i = start; i <= end; i++) {
      let numberOfDigits = i.toString().length;
      let sum = 0;
      let temp = i;
      while (temp > 0) {
          let remainder = temp % 10;
          sum += remainder ** numberOfDigits;
          temp = parseInt(temp / 10);
      }
      if (sum == i) {
          armstrongArray.push(i);
      }
  }
  return armstrongArray;
}

let getNextArmstrong = () => {
  let count = 0
  return {
      next: () => {
          let arr = getArmstrongNumbers(0,1000);
          return count < arr.length ?
              { armstrong_value: arr[count++], done: false } :
              { value: undefined, done: true }

      }
  }

}
const armstrongIterator = getNextArmstrong();
console.log(armstrongIterator.next());
console.log(armstrongIterator.next());
```

```
>> {"armstrong_value":0,"done":false}
>> {"armstrong_value":1,"done":false}
```

4. **Collections:** Using Set & Map, create a static data for chatting application. Here we have 2 chatrooms, every chatroom is having 3 users & every user has posted different messages in a chat room. Note that one user can belong to a single chat room only. Now you need to find out how you will hold this data using Set & Map data structures. Also add functionality to get fest of all users from a specific chatroom & listing down all message from a chatroom.

```
let chatroom1 = new Map();
chatroom1.set('Omkar', {
  msg1: "Hi everyone, I'm Omkar",
  msg2: "How are you all?"
});
chatroom1.set('Andrew', {
  msg1: "Hello, I'm Andrew",
  msg2: "I'm fine",
  msg3: "What do you do?"
});
chatroom1.set('Jhon', {
  msg1: "Hey, I'm Jhon",
  msg2: "I'm good",
  msg3: "I'm a Developer",
  msg4: "Let's develop an application..."
});

let chatroom2 = new Map()
  .set('Remo', {
    msg1: "Hey guys, I'm Remo",
    msg2: "I'm from UK",
    msg3: "Tell me about yours..."
  })
  .set('Sam', {
    msg1: "Hi, I'm Sam",
    msg2: "I am from US"
  });

// To check all the users in chatroom1
console.log("Total users in chatroom1: " + chatroom1.size);

// To list the users and messages
for (let [key, value] of chatroom1) {
  console.log(key, value);
}

// To check all the users in chatroom2
console.log("Total users in chatroom2 : " + chatroom2.size)
```

```javascript
chatroom2.forEach((value, key, map) => {
  console.log(`${key}`);
  console.log(chatroom2.get(`${key}`));
});

// To get the messages of specific user
console.log(chatroom1.get("Omkar"));

// To check user is in chatroom or not
console.log(chatroom1.has("Andrew"));

// To delete the user from chatroom
console.log(chatroom1.delete("Jhon"));

// To clear the chatroom
chatroom1.clear();
console.log("Total users in chatroom1: " + chatroom1.size);
```

```
>> Total users in chatroom1: 3
>> Omkar {"msg1":"Hi everyone, I'm Omkar","msg2":"How are you
all?"}
>> Andrew {"msg1":"Hello, I'm Andrew","msg2":"I'm
fine","msg3":"What do you do?"}
>> Jhon {"msg1":"Hey, I'm Jhon","msg2":"I'm good","msg3":"I'm
a Developer","msg4":"Let's develop an application..."}

>> Total users in chatroom2 : 2
>> Remo
>> {"msg1":"Hey guys, I'm Remo","msg2":"I'm from
UK","msg3":"Tell me about yours..."}
>> Sam
>> {"msg1":"Hi, I'm Sam","msg2":"I am from US"}

>> {"msg1":"Hi everyone, I'm Omkar","msg2":"How are you all?"}
>> true
>> true
>> Total users in chatroom1: 0
```