1) Find output

```
function outer() {
  var x = 10;

  function inner() {
    console.log(x);
    var x = 20;
  }

  return inner;
}

var closureFunc = outer();
closureFunc();
```

Ans) undefined

2) Find output

```
function createFunctions() {
  var result = [];

  for (let i = 0; i < 5; i++) {
    result.push(function() {
      console.log(i);
    });
  }

  return result;
}

var functions = createFunctions();
functions.forEach(fn => {
  fn();
})
```

Ans)

0
1
2
3
4

3) Implement a function that generates a sequence of unique IDs, starting from the given number

```
function createSequentialIdGenerator(baseValue) {
  // your code here
}

const generateUniqueId = createSequentialIdGenerator(999);

console.log(generateUniqueId()); // Expected output: 1000
console.log(generateUniqueId()); // Expected output: 1001
console.log(generateUniqueId()); // Expected output: 1002
```

Ans)

```
function createSequentialIdGenerator(baseValue) {
  var index = baseValue;

  function inner() {
        return ++index;
  }

  return inner
}

const generateUniqueId = createSequentialIdGenerator(999);

console.log(generateUniqueId()); // Expected output: 1000
console.log(generateUniqueId()); // Expected output: 1001
console.log(generateUniqueId()); // Expected output: 1002
```

4) Complete below code

```
function swapKeyAndValues(obj) {
  // Your code here
}

const sampleObject = {
  key1: 'value1',
  key2: 'value2',
  key3: 'value3'
};

swapKeyAndValues(sampleObject);
console.log(sampleObject);

// Expected output:
{
  value1: 'key1',
  value2: 'key2',
  value3: 'key3'
}
```

Ans)

```
function swapKeyAndValues(obj) {
  // Your code here
  for(key in obj) {
        obj[obj[key]] = key;
        delete(obj[key]);
  }
}

const sampleObject = {
  key1: 'value1',
  key2: 'value2',
  key3: 'value3'
};

swapKeyAndValues(sampleObject);
console.log(sampleObject);
```

5) Find whether all students in the class are passed in the exam
Rule: Passed - If average marks of a student > 40 else failed

```
  const students = [
  { name: 'John', marks: [70, 85, 90] },
  { name: 'Jane', marks: [60, 75, 80] },
  { name: 'David', marks: [50, 55, 65] }
];

function checkAllStudentsPassed(studentsArr) {
  // Your code here
}

const allStudentsPassed = checkAllStudentsPassed(students);

console.log(allStudentsPassed); // Output: true
```

Ans)

```
  const students = [
  { name: 'John', marks: [30, 25, 50] },
  { name: 'Jane', marks: [60, 75, 80] },
  { name: 'David', marks: [50, 55, 65] }
];

function checkAllStudentsPassed(studentsArr) {
  // Your code here
  result = studentsArr.every(
          (student) => {
          let sums = student["marks"].reduce(
          (total, current) => total+current, 0
          )
          return (sums/3)>40.0
          }
  )
  return result
}

const allStudentsPassed = checkAllStudentsPassed(students);

console.log(allStudentsPassed); // Output: true
```

6) Rewrite the below code snippet using async/await

```
function getProcessedData(url) {
      return downloadData(url)
      .catch(e => {
                  return downloadFallbackData(url)
      })
      .then(value => {
                  return processDataInWorker(value)
      })
}
```

Ans)

```
async function getProcessedData(url) {
 try {
          data = await downloadData(url)
 }
          catch(e) {
    return downloadFallbackData(url)
 }
 return processDataInWorker(value)
}
```

7) Implement Retry method using promise

```
function simulateAsyncTask() {
  return new Promise((resolve, reject) => {
    const randomNumber = Math.random();
    setTimeout(() => {
      if (randomNumber < 0.8) {
        resolve('Success');
      } else {
        reject('Error: Task failed');
      }
    }, 500);
  });
}
```

```
function retry() {
  // Your code here
}

// Sample invocation
retry(simulateAsyncTask, 3)
  .then(result => console.log('Result:', result))
  .catch(error => console.log('Error:', error));
```

Ans)

```
function retry(simulateAsyncTask, tries) {
        return simulateAsyncTask().catch(
        e => {
        if(tries>0) {
        console.log(tries)
        return retry(simulateAsyncTask, --tries);
        }
        throw e
        }
        )
}
```

8) Implement Retry method using async awai

Ans)

```
async function retry(simulateAsyncTask, tries) {
  try {
        return await simulateAsyncTask();
  }
  catch(error) {
        if(tries>0) {
        console.log(`attempt ${tries}`)
        return retry(simulateAsyncTask, --tries);
        }
        throw error
  }
}
```