1) Write a function that takes two numbers and returns their sum, ensuring the function is a pure function (i.e., no side effects).

```
function add(a, b) {
  return a + b;
}
```

2) Create a higher-order function that takes a function and an array as inputs, and applies the function to each element of the array.

```
function applyFunctionToArray(fn, arr) {
  return arr.map(fn);
}
```

3) Write a curried function that multiplies three numbers. The function should be curried such that you can call it one number at a time.

```
function multiply(a) {
  return function(b) {
    return function(c) {
      return a * b * c;
    };
  };
}
```

4) Create a function that takes an array and an element and returns a new array with the element added, ensuring the original array is not mutated.

```
function addToArray(array, element) {
  return [...array, element];
}
```

5) Write a function that returns a promise that resolves with a string "Success" after 1 second.

```
function delayedSuccess() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve("Success");
    }, 1000);
  });
}
```

6) Create an asynchronous function that fetches data from a mock API (simulated using setTimeout), and logs the result once it's fetched.

```
async function fetchMockData() {
  function mockAPI() {
```

```
    return new Promise(resolve => {
      setTimeout(() => {
        resolve({ data: "Mock data received" });
      }, 1000);
    });
  }

  const result = await mockAPI();
  console.log(result);
}
```

7) Write a function that takes a number and a callback function, applies the callback to the number, and returns the result.

```
function applyCallback(number, callback) {
  return callback(number);
}
```

8) Write a generator function that yields the first 3 numbers of the Fibonacci sequence.

```
function* fibonacciGenerator() {
  let a = 0, b = 1;
  yield a;
  yield b;
  yield a + b;
}
```

9) Create a Car class with properties make, model, and year, and a method displayInfo() that returns the car's information.

```
class Car {
  constructor(make, model, year) {
    this.make = make;
    this.model = model;
    this.year = year;
  }

  displayInfo() {
    return `Car: ${this.make} ${this.model}, Year: ${this.year}`;
  }
}
```

10) Create a Shape class with a method area(), and extend it into a Rectangle class that overrides the area() method to calculate the area of a rectangle.

```
class Shape {
  area() {
    return 0;
  }
}
```

```
class Rectangle extends Shape {
  constructor(width, height) {
    super();
    this.width = width;
    this.height = height;
  }

  area() {
    return this.width * this.height;
  }
}
```