Closure

1. Create a counter function using closure that increments a count every time it's called.

```
const counter = createCounter();
console.log(counter()); // Output: 1
console.log(counter()); // Output: 2
```

---

2. Implement a function that returns an object with methods to get and set a private variable.

```
const secretValue = privateVariable(42);
console.log(secretValue.get()); // Output: 42
secretValue.set(100);
console.log(secretValue.get()); // Output: 100
```

---

3. Write a function that caches the duplication of result of a calculation using closure.

```
const cachedCalculation = createCachingFunction();
console.log(cachedCalculation(5)); // Output: 10 (calculated)
console.log(cachedCalculation(5)); // Output: 10 (cached)
```

---

4. Implement a simple person object with private properties using closures.

```
const person = createPerson("Alice", 30);
console.log(person.getName()); // Output: "Alice"
console.log(person.getAge()); // Output: 30
person.setName("Bob");
person.setAge(25);
console.log(person.getName()); // Output: "Bob"
console.log(person.getAge()); // Output: 25
```

---

5. Write a function that received callback and set of arguments and calculate the result via callback using closure.

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

```
const add5 = partial(add, 5);
console.log(add5(10, 20)); // Output: 35 (5 + 10 + 20)
```

---

6. Implement a memoization function using closure to cache expensive function calls.

```
function fibonacci(n) {
  ...
}

const memoizedFibonacci = memoize(fibonacci);
console.log(memoizedFibonacci(10)); // Output: 55 (calculated)
console.log(memoizedFibonacci(10)); // Output: 55 (cached)
```

---

7. Create a function factory that generates functions for different operations.

```
const add = createCalculator("add");
console.log(add(3, 5)); // Output: 8

const multiply = createCalculator("multiply");
console.log(multiply(2, 4)); // Output: 8
```

---

8. Implement a currying function using closures.

```
const curriedAdd = curry(add);
console.log(curriedAdd(1)(2)(3)); // Output: 6
console.log(curriedAdd(1, 2)(3)); // Output: 6
console.log(curriedAdd(1, 2, 3)); // Output: 6
```