Definição de Função como Retorno

```
// Note que desagregamos os parâmetros.
const potencia = (base) => (expoente) => base ** expoente;
// Passamos 5 como argumento para o parâmetro base e 3 como
// argumento para o parâmetro expoente.
const respostaUm = potencia(5)(3);
console.log(`O resultado é: ${respostaUm}`);

const respostaDois = potencia(81)(1/2);
console.log(`O resultado é: ${respostaDois}`);

const respostaTres = potencia(2)(-1);
console.log(`O resultado é: ${respostaTres}`);


// [EXEMPLO] Defina funções para calcular o quadrado, o cubo e a
// raiz quadrada de um número passado
// como argumento reaproveitando uma definição de função genérica chamada expoente.


// Passamos primeiro o expoente que queremos.
const expoente = (expoente) => (base) => base ** expoente;
// Retorna (base) => base ** 2
const quadrado = expoente(2);
// Retorna (base) => base ** 3
const cubo = expoente(3);
// Retorna (base) => base ** 1/2
const raizQuadrada = expoente(1/2);

console.log(`\nO resultado é: ${quadrado(10)}`);
console.log(`\nO resultado é: ${cubo(3)}`);
console.log(`\nO resultado é: ${raizQuadrada(81)}`);
```

Definição de Função como Argumento

```
// Definição de função como argumento

// Se uma função é um valor então temos que ela
// pode ser passada como argumento para outras funções.

// Note que declaramos várias funções básicas
const subtrair = (x, y) => x - y;
const somar = (x, y) => x + y;
const multiplicar = (x, y) => x * y;
const divir = (x, y) => x / y;
const concatenar = (x, y, sep=' ') => x+sep+y;
const iniciais = (x, y) => x[0] + y[0];
// const negativo = subtrair(0, y);

// Note que temos um f ali entre parênteses, isso significa
// que estamos passando uma função como argumento para outra função.
const exec = (f, x, y) => f(x, y);

// Passando função como argumento para a função exec
// Chama a função subtrair e passa 50, 25 como argumento para x, y.
const resposta1 = exec(subtrair, 50, 25);

const resposta2 = exec(somar, 50, 25);

const resposta3 = exec(multiplicar, 50, 25);

const resposta4 = exec(divir, 50, 25);

const resposta5 = exec(concatenar, 'Isaac', 'Newton');

const resposta6 = exec(iniciais, 'Isaac', 'Newton');

// Note que todas as funções acima receberam uma outra função como argumento.
console.log(`\nO resultado é: ${resposta1}`);
console.log(`O resultado é: ${resposta2}`);
console.log(`O resultado é: ${resposta3}`);
console.log(`O resultado é: ${resposta4}`);
console.log(`O resultado é: ${resposta5}`);
console.log(`O resultado é: ${resposta6}`);
```

No exemplo acima temos que ao chamarmos a função resposta1 ela irá chamar a funçao exec() que realiza a chamada da função passada como argumento, nesse caso subtrair() e logo em seguida passamos 50 e 25 como argumentos para os parêmtros x e y. Após isso, a função resposta1 retorna uma outra função que é a função subtrair, mas antes ela realiza a chamada da função subtrair que executa sua tarefa. Por fim, a função exec retorna o valor de sua operação.

Função Anônima

```
// Funções anônimas

// Podemos definir uma função qualquer apenas no momento de seu uso, ou seja
// não é necessário atribuir um nome a ela.

// [EXEMPLO: Lista 02, Q12] Escreva um programa para calcular o maior
// e o menor valor real das raízes de uma equação de segundo grau.

// Temos os três argumentos a b c e retornamos uma função.
const raiz = (a, b, c, f) => {
    // Realiza o cálculo do delta.
    const delta = (b ** 2) - (4 * a * c);
    // Faz uma análise do delta caso negativo.
    if (delta < 0) return undefined;

    // Armazena os valores das raízes.
    const positivo = ((-b) + delta) / (2 * a);
    const negativo = ((-b) - delta) / (2 * a);
    // Retorna uma função que foi passada como argumento.
    return f(positivo, negativo);
}

// Valores como argumento
const a = 1;
const b = -5;
const c = 6;

// Chamamos a função e realizamos a checagem das raízes.
// x e y são as raízes que são retornadas da função.
const maiorRaiz = raiz(a, b, c, (x, y) => (x >= y? x : y));
const menorRaiz = raiz(a, b, c, (x, y) => (x <= y? x : y));

// Realiza a checagem das raízes.
const textoSaida = (x, y) => x == undefined?
 'Não há raízes reais!' : `As raízes da equação são: ${x} e ${y}!`;

// Chama a função textoSaida que recebe dois parâmetros(as raízes) e exibe o resultado.
// As raízes da equação são: 2 e 3!
console.log(textoSaida(menorRaiz, maiorRaiz))
```

No exemplo acima temos que a função raiz recebe uma outra função como argumento. Além disso, essa função que é recebida como argumento é uma função que será retornada no fim da nossa função raiz. Em seguida declaramos uma função anônima dentro da chamada da função raiz nas variáveis maiorRaiz e menorRaiz. Note que a função anônima realiza uma checagem das raízes(positivo, negativo) e armazena os seus valores.

Lista 03

```
// Q1. Programa para calcular a área de um retângulo dados os
//comprimentos de duas arestas

// Note que separamos os parâmetros.
const areaRetangulo = (base) => (altura) => base * altura;

const base = 2;
const altura = 3;

console.log(`Base: ${base}, Altura: ${altura}, Área: ${areaRetangulo(base)(altura)}`);
```

No exemplo acima separamos os parâmetros para retornar expressões como valor. Note que na chamada da função areaRetangulo temos que a primeira chamada irá retornar uma expressão(base) => (altura) => base * altura; e na chamada seguinte, o resultado anterior será calculado.

```
// Q2. Programa para calcular a área de uma circunferência dado o valor do raio.

const areaCircunferencia = (raio) => (pi = 3.1415) => raio ** 2 * pi;

const raio = 5;

// Predefinimos pi como sendo 3.1415 mas devemos passar um parênteses vazio.
console.log(areaCircunferencia(raio)());
```

```
// Q3. Programa para determinar se três valores passados podem
// representar um triângulo ou não.

const representaTriangulo = (a) => (b) => (c) => {
    if (((a + b) > c) && ((a + c) > b) && ((b + c) > a)) {
        return true;
    } else {
        return false;
    }
}

const a = 16;
const b = 20;
const c = 30;

const resultado = representaTriangulo(a)(b)(c);

// Essa função recebe um valor true ou false.
const funcTeste = (tOrF) => tOrF == true? '' : 'não ';

// Exibimos o resultado chamando a função funcTeste para verificar o resultado.
const displayResult =
`Os lados ${a}, ${b} e ${c} ${funcTeste(resultado)}podem formar um triângulo!`;

console.log(displayResult);
```

No exemplo acima usamos uma outra função que recebe um valor booleano(true ou false) e retorna o resultado da condição.

```
// Q4. Programa para classificar um triângulo em Equilátero
// Isósceles ou Escaleno a partir
// dos valores de seus três lados.

const representaTriangulo = (a) => (b) => (c) => {
    if (((a + b) > c) && ((a + c) > b) && ((b + c) > a)) {
        return true;
    } else {
        return false;
    }
}

const a = 2;
const b = 2;
const c = 2;

// Armazena true ou false.
const resultado = representaTriangulo(a)(b)(c);

// Recebe um valor booleano e os três lados do triângulo
const tipoTriangulo = (r) => (a) => (b) => (c) => {
    // Analisa o valor de r, caso seja true o código abaixo é executado.
    if (r) {
        if (a == b && a == c) {
            return `Equilátero!`;
        } else if (a == b || a == c) {
            return `Isósceles!`;
        } else {
            return `Escaleno`;
        }
    } else {
        return `Não forma triângulo!`;
    }
};

console.log(tipoTriangulo(resultado)(a)(b)(c));
```

```
// Q5. Programa para calcular a distância euclidiana entre dois pontos
// [(x1, y1), (x2,y2)][(x1,y1),(x2,y2)] no plano cartesiano.
// Utilize a equação geral da reta para calcular a distância quando
// a reta não for paralela a nenhum dos eixos (abscissas ou ordenadas)
// e as versões simplificadas quando for paralela.

const distanciaDoisPontos = (x1) => (y1) => (x2) => (y2) => {
    // Armazenamos a diferença entre os pontos.
    const pX1X2 = Math.abs(x1 - x2);
    const pY1Y2 = Math.abs(y1 - y2);

    // Paralelo a Ordenada.
    if (x1 == x2) return pY1Y2;
    // Paralelo a Abscissa.
    else if (y1 == y2) return pX1X2;
    // Retorna o resultado da distância entre dois pontos.
    else return Math.sqrt(pX1X2 ** 2 + pY1Y2 ** 2);
};

// Pontos x e y
const pX1 = 3;
const pY1 = 4;
const pX2 = 12;
const pY2 = 0;

// Chamamos a função e passamos os respectivos valores.
const exibeResultado = distanciaDoisPontos(pX1)(pY1)(pX2)(pY2);
// Mostramos o resultado.
console.log(`A distância entre os pontos é: ${exibeResultado}!`);
```

```
// Q6. Fornecidos três valores, a, b e c, escreva um programa que retorne
// quantos dos três são iguais. A resposta pode ser 3 (todos iguais)
// 2 (apenas um diferente) ou 0 (todos diferentes).

// Testamos todos os 3 casos e retornamos o número de valores iguais.
const qntsValores = (a) => (b) => (c) => {
    if (a == b && a == c) return 3;
    else if (a == b || b == c || a == c) return 2;
    else return 0;
};

const x = 6;
const y = 0;
const z = 5;

// Armazenamos o resultado
const exibeResultado = qntsValores(x)(y)(z);

// Exibimos o número de valores iguais.
console.log(`${exibeResultado} valores iguais!\n`);


// Q7. Programa para retornar o menor valor entre três números quaisquer.
// Tente resolver considerando o sub-problema de determinar o menor valor
// entre dois números quaisquer (obs: em caso de valores iguais, deve-se
// retornar como resultado o próprio valor).

const menorValor = (x) => (y) => {
    // Retorna o menor valor.
    if (x >= y) return y;
    else return x;
};

// Declaramos 3 valores
const x = -3;
const y = -5;
const z = -100;

// Note que não precisamos criar uma outra função para comparar os 3 valores
// Basta reutilizar a função passando um valor e depois mais dois
// outros valores(y e z) para realizar a comparação.
const resultadoMenorValor = menorValor(x)(menorValor(y)(z));

// Menor valor = -100.
console.log(`O menor valor entre ${x}, ${y} e ${z} é: ${resultadoMenorValor}`);
```

```
// Q8. Escrever um programa que calcule o valor de um número elevado à quarta potência.
// Tente fazer uso do sub-problema de calcular o quadrado de um número qualquer.

// Para resolver essa questão basta criar uma função qualquer que recebe
// primeiro o EXPOENTE e depois a base.

const potenciaNumero = (expoente) => (base) => expoente ** base;

const expQuatro = 4;

// Passamos o expoente 2
// Armazenamos (base) => 4 ** base.
const quartaPotencia = potenciaNumero(expQuatro);

// Exibe o resultado.
console.log(`A quarta potência do valor ${2} é: ${quartaPotencia(2)}`);
```

No exemplo acima quando nós chamamos a função quartaPotencia no console.log e passamos 2 como argumento, temos que a função potenciaNumero será chamada e o valor que está armazenado na variável quartaPotencia((base) => base ** 4) será calculado pela função potenciaNumero, que retorna 2 ** 4 = 16.

```
// Q9. Programa que calcula o "ou-exclusivo" entre dois
// valores-verdade (verdadeiro ou falso).

const ouExclusivo = (a) => (b) => {
    // Definição matemática de ou exclusivo.
    return (a || b) && !(a && b);
};

let a = true;
let b = false;

// Retorna true.
console.log(`a = ${a} e b = ${b}, XOR = ${ouExclusivo(a)(b)}`)

// Testando para caso os dois valores sejam true e false ao mesmo tempo.

// Reatribuindo os valores.
a = true;
b = true;

// Retorna false.
console.log(`a = ${a} e b = ${b}, XOR = ${ouExclusivo(a)(b)}`)

// Reatribuindo os valores.
a = false;
b = false;

// Retorna false.
console.log(`a = ${a} e b = ${b}, XOR = ${ouExclusivo(a)(b)}`)


// Q10. Escreva um programa que dados o primeiro nome e o último sobrenome de
// uma pessoa qualquer, retorne-os em forma de citação bibliográfica conforme o
// exemplo: Fulano Santos --> Santos, Fulano.

// Basta retornar uma string literal com o sobrenome ficando em primeiro.
const citacao = (nome) => (sobrenome) => `${sobrenome}, ${nome}.`;

// Declaração e atribuição de variáveis.
const nome = "Douglas";
const sobrenome = "Cunha";

// Chamamos a função e passamos os respectivos valores.
console.log(citacao(nome)(sobrenome));
```

```
// Q11. Dados três valores, escreva um programa que retorne quantos
// desses três valores são maiores que o valor médio entre eles.

// Primeiro calculamos a média.
const mediaValores = (a) => (b) => (c) => {
    // Retornamos a média entre os 3 argumentos.
    return (a + b + c) / 3;
};

// Depois criamos uma função que irá analisar quantos valores são maiores.
const qntsMaiores = (x) => (y) => {
    // Realizamos a checagem.
    if (x > y) {
        return 1;
    } else {
        return 0;
    }
};

// Declaramos variáveis e atribuimos os valores.
const a = 7;
const b = 7;
const c = 9;

// Chamamos a função mediaValores e armazenamos o seu resultado.
const mediaTotal = mediaValores(a)(b)(c);

// Agora nós iremos chamar a função qntsMaiores e passar cada argumento de forma
// índividual e em seguida, chamamos a função mediaTotal e comparamos os valores
// sendo a = x e mediaTotal = y.

// Exemplo: 0 + 0 + 1 = 1.
const resultado =
qntsMaiores(a)(mediaTotal) + qntsMaiores(b)(mediaTotal) + qntsMaiores(c)(mediaTotal);

console.log(`Os valores são: a = ${a}, b = ${b} e c = ${c}`);
// Somente o 9 é maior que a média(7.6), logo 1 resultado.
console.log(`Total de números maiores que a média: ${resultado}`);
```

```
// Q12. Escreva um programa para calcular o maior e o menor valor real
// das raízes de uma equação de segundo grau.

// Criamos uma função que recebe 3 valores e um valor que índica a
// raiz positiva ou negativa. .
const resultadoRaiz = (a) => (b) => (c) => (pOuN) => {
    // Calculamos o delta
    const delta = (b ** 2) - 4 * a * c;

    // Calcula e armazenamos as raízes
    const raizPositiva = ((-b) + Math.sqrt(delta)) / (2 * a);
    const raizNegativa = ((-b) - Math.sqrt(delta)) / (2 * a);

    // Testamos para o parâmetro pOuN
    if (pOuN > 0) return raizPositiva;
    else return raizNegativa;
};

// Declaramos três variáveis e atribuimos valores.
const a = -2;
const b = 1;
const c = 3;

// Criamos duas variáveis e passamos os valores de a, b, c e qual
// raiz queremos, ou seja, positiva ou negativa.
const raizN = resultadoRaiz(a)(b)(c)(-1);
const raizP = resultadoRaiz(a)(b)(c)(1);

// Exibimos o resultado.
console.log(`As raízes da equação são dadas por: ${raizN} e ${raizP}`);


// Q13. Um móvel com velocidade constante percorre uma trajetória retilínea.
// Considere t0 = 0, o instante inicial e x0 = 500 a posição inicial.
// Escreva um programa para calcular a velocidade do
// objeto em um dado instante t e posição x.

const velocidade = (t) => (x) => (t0 = 0) => (x0 = 0) => {
    // Posição final menos inicial / Tempo final - inicial.
    return (x - x0) / (t - t0);
};

// Variáveis e atribuições.
const t0 = 0;
const x0 = 500;
const t = 30;
const x = 2000;

// Resultado = 50.
console.log(velocidade(t)(x)(t0)(x0));
```

Nota: A questão 14 foi pulada por motivos de que ela representa apenas algarismo de 0 até 9.