Fáceis:

1) O que caracteriza uma função pura?
A) Ela sempre retorna o mesmo resultado para os mesmos argumentos.
B) Ela modifica o estado de variáveis globais.
C) Ela depende de variáveis externas.
D) Ela sempre executa um loop.
RESPOSTA: A

2) Qual é o resultado da seguinte função pura:
```
const dobro=(x)=> {return x * 2}
```
quando chamada com o argumento 4?
A) 2
B) 4
C) 8
D) 10
RESPOSTA:C

3) Assinale a alternativa que NÃO é uma característica de funções puras:
A) Imutabilidade de dados.
B) Dependência de estado externo.
C) Consistência em resultados.
D) Reusabilidade
RESPOSTA:B

4) Qual dos seguintes exemplos representa uma violação do conceito de função pura?
A) `const quadrado=(x)=> {return x * x}`
B) `const adicionaTempo=(data, dias)=> {return data + timedelta(days=dias)}`
C) `const multiplica=(x, y)=> {return x * y}`
D) `const geraId=()=> return random.randint(1, 100)`
RESPOSTA:D

5) Assinale a alternativa que completa corretamente o cálculo do volume de uma esfera, dada pela fórmula V= 4/3 x r³
function v(r, ___){
*return(4/3 * pi * ___)*
A)3.14, r**3
B) pi=3.14, r*3

C) pi=3.14, r**3
D) 3.14, r*3
RESPOSTA:C

6) Funções puras facilitam o uso de qual técnica?
A) Programação orientada a objetos.
B) Testes unitários.
C) Programação assíncrona.
D) Manipulação de estado.
RESPOSTA:B

7) Assinale a alternativa que representa uma função pura:
A) `const soma=(x, y)=>{ return x + y + z }`*(onde z é uma variável global)*
B) `const soma=(x, y)=> {return x + y}`
C) `const incrementa=(x)=> {x + y + 1}`*(onde y é uma variável global)*
RESPOSTA:B

8) Qual das seguintes operações é considerada um efeito colateral?
A) Retornar o dobro de um número.
B) Modificar uma variável global.
C) Somar dois números.
D) Calcular a raiz quadrada de um número
RESPOSTA:B

9) Qual das alternativas abaixo é um benefício de usar funções puras? A) Aumento da complexidade do código.
B) Facilidade para testar e depurar.
C) Menor legibilidade.
D) Dependência de estado global.
RESPOSTA:B

10) Assinale o comportamento da função abaixo

```
const dividir = (x,y) =>{
 if (y==0) return "ERRO"
 return x/y
```

A) Retorna erro
B) Multiplica *x* por *y*
C) Divide *x* por *y*
D) Eleva *x* a *y*

RESPOSTA:C

Médias:

1) Considere a função `const double=(x)=>{ return 2 * x}`. Qual é o efeito colateral dessa função?

A) Nenhum, é uma função pura.
B) Modifica o valor de x.
C) Afeta uma variável global.
D) Imprime o valor de x.

RESPOSTA: A

2) Em uma linguagem funcional, como a imutabilidade se relaciona com funções puras?

A) Funções puras não podem trabalhar com dados mutáveis.
B) A imutabilidade não afeta a pureza das funções.
C) Funções puras exigem dados mutáveis para serem eficientes.
D) A imutabilidade facilita a criação de funções puras.
RESPOSTA: D

3) Se uma função precisa acessar dados de configuração externa, como isso deve ser feito para mantê-la pura?

A) Acessando diretamente a configuração dentro da função.
B) Passando a configuração como um parâmetro para a função.
C) Armazenando a configuração em uma variável global.
D) Modificando a configuração durante a execução da função.

RESPOSTA: B

4) Quando uma função pura é chamada várias vezes com os mesmos argumentos, o que acontece?

A) Ela pode retornar resultados diferentes.

B) O resultado é sempre o mesmo.

C) Ela provoca efeitos colaterais diferentes a cada chamada.

D) Ela consome mais memória a cada chamada.

RESPOSTA: B

5) Dada a seguinte função

function concatStrings(a, b) {

return a + b; }

Qual dos seguintes exemplos de chamada mantém a pureza da função?

A) `concatStrings("Hello, ", "world!")`
B) `let str = "Hello, "; concatStrings(str, "world!")`
C) `str += "updated"; concatStrings("Hello, ", str)`
D) Todas as anteriores

RESPOSTA: D

6) Considere a seguinte função

function doubleArray(arr) {

return arr.map(num => num * 2); }

Essa função é pura ou impura?

A) Pura
B) Impura

RESPOSTA: A

7) Considere a seguinte função

function multiplyByTwo(num, multiplier=2) {

*return num * multiplier}*

Qual é o resultado ao chamar `multiplyByTwo(3)`*?*

A) 3

B) 4

C) 2

D) 6

RESPOSTA: D

8) Dado o código abaixo, qual é a saída da função `calculate`?

```
function calculate(a, x=10) {
    return a + x}
console.log(calculate(5))
```

A) 5

B) 10

C) 15

D) 20

9) No contexto de programação funcional, qual dos seguintes conceitos está mais relacionado à utilização de funções puras?

A) Efeitos colaterais
B) Mutabilidade de estado
C) Imutabilidade
D) Execução sequencial

RESPOSTA: C

10) Dado o seguinte código, qual é a melhor forma de torná-lo uma função pura?

```
const total = 20
function addToTotal(num) {
   return total + num
```

A) Fazer a função retornar `num`.

B) Passar `total` como argumento e retornar a soma.

C) Utilizar uma variável global para armazenamento.

RESPOSTA: C

Difíceis:

1) Considere a seguinte função em uma linguagem funcional:

   function filterEven(numbers):

   return [n for n in numbers if n % 2 == 0]

Essa função é considerada pura? Justifique.

A) Sim, porque não altera a lista original de `numbers`.

B) Não, porque depende de uma variável externa.

C) Sim, porque sempre retorna uma nova lista.

D) Não, porque a operação de filtro pode causar efeitos colaterais.

RESPOSTA: A

2) Qual é o impacto do uso de funções puras na implementação de algoritmos recursivos?

   A) Aumenta a chance de estouro de pilha devido a chamadas recursivas profundas.

   B) Permite otimizações como Tail Call Optimization.

   C) Reduz a legibilidade do código.

   D) Exige a utilização de variáveis globais para controle de estado.

   RESPOSTA: B

3) Qual é a principal implicação de usar funções puras em um sistema de múltiplas threads?

   A) Aumento do uso de locks para garantir segurança.

   B) Redução de problemas de concorrência e condições de corrida.

   C) Necessidade de implementação de um sistema de estado compartilhado.

   D) Aumento da complexidade do gerenciamento de memória.

   RESPOSTA: B

4) Em programação funcional, como a imutabilidade se relaciona com a definição de funções puras?

   A) A imutabilidade é irrelevante para funções puras.

   B) Funções puras dependem de variáveis mutáveis.

   C) A imutabilidade garante que funções não causarão efeitos colaterais.

   D) Funções puras não podem manipular dados imutáveis.

   RESPOSTA: C

5) Em um contexto de programação funcional, o que significa a expressão "referência transparente"?

   A) Uma expressão que sempre pode ser substituída por seu valor sem alterar o comportamento do programa.

   B) Uma expressão que não pode ser usada em funções puras.

   C) Uma expressão que causa efeitos colaterais em variáveis globais.

   D) Uma expressão que não pode ser otimizada pelo compilador.

   RESPOSTA: A

6) Qual dos seguintes padrões de design é frequentemente utilizado em conjunto com funções puras para melhorar a composição de funções?

   A) Padrão de Singleton

   B) Funções de ordem superior

   C) Padrão Observer

   D) Padrão de Comando

   Resposta**:** B

7) Por que o conceito de memoization é particularmente relevante para funções puras?

   A) Porque funções puras não podem ser otimizadas.

   B) Porque memoization é uma forma de alterar o estado global.

C) Porque memoization pode melhorar o desempenho sem causar efeitos colaterais.

D) Porque memoization é irrelevante para funções que dependem de variáveis externas.

Resposta**:** C

8) Como as funções puras podem influenciar a arquitetura de um sistema em microserviços?

A) Elas permitem o compartilhamento de estado entre serviços.

B) Elas facilitam a testabilidade e a previsibilidade dos serviços.

C) Elas aumentam a complexidade da comunicação entre serviços.

D) Elas requerem o uso de bancos de dados mutáveis.

9) Em uma linguagem que suporta programação funcional, como a imutabilidade afeta a eficiência de funções puras?

A) Imutabilidade pode levar a cópias desnecessárias de dados, diminuindo a eficiência.

B) Imutabilidade não tem impacto na eficiência.

C) Imutabilidade sempre aumenta a eficiência de funções puras.

D) Funções puras são sempre ineficientes por causa da imutabilidade.

RESPOSTA: A

10) Qual é o papel das funções de ordem superior na implementação de funções puras?

A) Elas são necessárias apenas para funções impuras.

B) Elas permitem a criação de funções que podem manipular outras funções como argumentos.

C) Elas não têm relação com a pureza das funções.

D) Elas aumentam a complexidade da implementação de funções puras.

RESPOSTA: B