Desafio: Sequência de Fibonacci Recursiva

A sequência de Fibonacci é uma série de números onde cada número é a soma dos dois anteriores. A sequência começa com 0 e 1. Ou seja:

- F(0) = 0
- F(1) = 1
- F(n) = F(n-1) + F(n-2) para n > 1

Crie uma função recursiva que calcule o n-ésimo número de Fibonacci. Depois, escreva um programa que peça para o usuário inserir um número inteiro (n) e imprima o valor de F(n).

Exemplo de Entrada e Saída:

Entrada:

In [ ]:
```
Digite um número: 6
```

Saída:

In [ ]:
```
O número Fibonacci na posição 6 é: 8
```

In [ ]:
```
```

In [1]:
```
fibonacci <- function(number) {
    if (number == 0 | number == 1) {
        return(number)
    } else {
        return(fibonacci(number - 1) + fibonacci(number - 2))
    }
}
```

In [2]:
```
# Solicitar um número ao usuário
cat("Digite um número: ")
n <- as.integer(readline())

# Calcular e exibir o resultado
cat(paste("O número Fibonacci na posição", n, "é:", fibonacci(n), "\n"))
```

```
Digite um número:
O número Fibonacci na posição 6 é: 8
```

In [ ]:
```
```

In [3]:
```
# Criar uma função com memoization
fibonacci <- local({
    # Ambiente para armazenar valores já calculados
    cache <- new.env()
```

```
    function(number) {
        if (exists(as.character(number), envir = cache)) {
            # Retorna o valor do cache se já calculado
            return(get(as.character(number), envir = cache))
        }

        # Calcular o valor de Fibonacci
        result <- if (number == 0 | number == 1) {
            number
        } else {
            fibonacci(number - 1) + fibonacci(number - 2)
        }

        # Armazenar o valor no cache
        assign(as.character(number), result, envir = cache)
        return(result)
    }
})

# Solicitar um número ao usuário
cat("Digite um número: ")
n <- as.integer(readline())

# Calcular e exibir o resultado
cat(paste("O número Fibonacci na posição", n, "é:", fibonacci(n), "\n"))
```

```
Digite um número:
O número Fibonacci na posição 78 é: 8944394323791464
```

In [ ]:

Explicação:

1. **Cache**: Um ambiente (`cache`) é usado para armazenar os valores de Fibonacci já calculados, onde a chave é o número (`number`) e o valor é o resultado.
2. **Memoization**: Antes de calcular o Fibonacci de um número, a função verifica se ele já foi calculado e está armazenado no cache.
3. **Eficiência**: Isso reduz drasticamente o tempo de execução, especialmente para valores grandes de `n` .