# Paradigma de Programação Funcional

## 5 - Combinação de templates - Exercícios

**5.1)** Utilizando apenas as funções primitivas `zero?`, `add1` e `sub1`, escreva as funções >, >=, <, <= e =. Cada função deve receber como parâmetro dois números naturais e executar a operação de comparação apropriada.

**5.2)** Defina uma função que devolve os $n$ primeiros elementos de uma lista.

```
> (take (list 10 40 70 20 3) 2)
'(10 40)
```

**5.3)** Defina uma função que devolve um nova lista sem os $n$ primeiros elementos de uma dada lista.

```
> (drop (list 10 40 70 20 3) 2)
'(70 20 3)
```

**5.4)** Defina uma função que devolva uma nova lista que é como a lista de entrada mas sem o elemento de uma dada posição.

```
> (remove-at (list 3 6 1 2) 2)
'(3 6 2)
```

**5.5)** Defina uma função que devolva uma lista que é como a lista de entrada mas com um dado elemento inserido em uma dada posição.

```
> (insert-at (list 3 6 1 2) 5 2)
'(3 6 5 1 2)
```

**5.6)** [pp99 1.18] Defina uma função que devolva uma sub-lista de uma lista com um intervalo especificado.

```
> (sub-list (list 10 20 30 40 50) 1 4)
'(20 30 40)
```

**5.7)** [pp99 1.16] Defina uma função que devolva uma nova lista que é como a lista de entrada mas com os elementos rotacionados $n$ posições a esquerda.

```
> (rotate-left (list 10 20 30 40 50) 2)
'(30 40 50 10 20)
```

**5.8)** Dado duas listas de entrada, `ltsa` e `ltsb`, defina uma função que devolva uma nova lista com os elementos de `ltsa` seguidos dos elementos de `ltsb`.

```
> (append (list 3 7 12) (list 2 4 5))
'(3 7 12 2 4 5)
```

**5.9)** Dado duas listas de números em ordem crescente, defina uma função que devolva uma nova lista com os elementos das duas listas de entrada em ordem crescente.

```
> (merge (list 3 7 12) (list 2 4 5))
'(2 3 4 5 7 12)
```

## Referências


- [pp99]. 99 problemas para resolver em (Prolog) Racket

# Licença