Lista 2 de LP

1. Escreva, usando *foldl* ou *foldr* uma função que recebe uma lista de cadeias de caracteres (valores do tipo *String*) e retorna uma cadeia de caracteres que contém os 3 primeiros caracteres de cada cadeia.

   Por exemplo, ao receber `["Abcde", "1Abcde", "12Abcde", "123Abcde"]` deve retornar `"Abc1Ab12A123"`.

2. Escreva, usando *foldr* ou *foldl*, uma função que recebe uma lista de valores do tipo *Pessoa*, definido a seguir, e retorna a soma das idades (valores do campo *idade*) de todos os elementos da lista.

```
data Pessoa = Pessoa {nome::Nome, idade::Idade, id::RG}
type Nome   = String
type Idade  = Integer
type RG     = String
```

   Note: a definição do tipo *Pessoa* com campo *idade* de tipo *Idade* cria automaticamente função *idade*:: *Pessoa* -> *Idade*, que seleciona o valor no campo *idade* do argumento de tipo *Pessoa*.

3. Escreva, usando *foldr* ou *foldl*, uma função que recebe uma lista não vazia de valores de tipo *Pessoa*, da definição acima, e retorna o nome da pessoa mais nova da lista.
4. Escreva, usando *foldl* ou *foldr*, uma função que recebe uma lista de cadeias de caracteres (valores do tipo *String*) e retorna uma cadeia de caracteres que contém os 3 primeiros caracteres de cada cadeia removidos se não forem letras, ou com as letras em caixa alta se forem letras, e com os demais caracteres depois dos 3 primeiros sem alteração.

   Por exemplo, ao receber `["Abcde", "1Abcde", "12Abcde", "123Abcde"]` deve retornar `"ABCdeABcdeAbcdeAbcde"`

5. Explique porque foldr f x pode não percorrer toda a lista x, ao passo que toda a lista x é sempre percorrida, no caso de foldl.
6. A função *remdups* remove elementos iguais adjacentes de uma lista, conservando só um dos elementos.

   Por exemplo, *remdups* `[1,2,2,3,3,3,1,1] = [1,2,3,1]`.

   Defina *remdups* usando *foldr* ou *foldl*.