*(Continuação da parte I, pode usar o PC)*

1. *Considere os números de Romano, definidos da seguinte forma:* `z` *denota zero,* `i(X)` *denota* `X+1`*,* `v(X)` *denota* `X+5`*,* `x(X)` *denota* `X+10`*,* `l(X)` *denota* `X+50`*,* `c(X)` *denota* `X+100`*,* `d(X)` *denota* `X+500` *e* `m(X)` *denota* `X+1000`*. A notação é sempre* ***decrescente****, i.e. o número 23 deverá ser escrito* `x(x(i(i(i(z)))))`*, nunca* `x(i(x(i(i(z)))))`*, por exemplo.*

   (a) *Defina um predicado de tipo* `romano/1` *que sucede se o seu argumento for um número de Romano válido,* sem exigir a ordem decrescente.

   (b) *Redefina o predicado* `romano/1` *para só admitir números que satisfaçam a ordem dos componentes (i.e. ordem decrescente, como no enunciado.)*

   (c) Defina um predicado `romano_desfaz/2` que sucede se o seu primeiro argumento for um número de Romano **válido** e o segundo argumento for o número inteiro correspondente. Assuma que o número de Romano está instanciado (i.e. encontre o valor de um número de Romano.)

```
| ?- romano_desfaz(x(x(i(i(i(z))))), X).
X = 23
```

   Defina um predicado `romano_faz/2`, igual ao anterior mas que se comporta de forma inversa: dado um número inteiro, vamos produzir o número de Romano correspondente.

```
| ?- romano_faz(R, 23).
R = x(x(i(i(i(z)))))
```

2. Queremos trabalhar com números de Romano em Ocaml. Considere que temos a seguinte declaração de tipo:

```
type romano =
  | Z
  | I of romano | V of romano | X of romano | L of romano
  | C of romano | D of romano | M of romano;;
```

   (a) Defina uma função `valor` que retorne o `int` que é o valor dum `romano`; por exemplo queremos que:

```
# valor Z;;
- : int = 0
# valor (X (X (I (I (I Z)))));;
- : int = 23
```

(b) Defina outra função, `soma`, que retorne a soma dos seus dois argumentos, sendo tudo expresso como números de Romano. Por exemplo:

```
# let rec soma x y = (* definição ... *) ;;
val soma : romano -> romano -> romano = <fun>
# soma (X(I(Z))) (L(I(Z)));;
- : romano = L (X (I (I Z)))
# soma (X(X(X(Z)))) (X(X(X(V(Z)))));;
- : romano = L (X (V Z))
```

(c) Redefina o tipo `romano` para só permitir números de Romano válidos, i.e. por ordem decrescente, p/ex devemos poder escrever `X(V(I(I(Z))))` mas não `V(I(X(I(Z))))`: o primeiro caso dará um valor enquanto o segundo dará um erro de sintaxe.