Teste 2022

Resolva:

- Dois dois exercícios 1 a 3, à sua escolha (2 x 8 = 16 valores)
- O exercício 4 (4 valores)

1. Entregue as suas propostas de solução em papel
2. Teste as suas propostas em casa (copie tudo o que fizer) e envie por email um pequeno relatório comentando o que teria de ser alterado no que entregou, por forma a se obter implementações verificadas das funções pedidas.

Exercício 1

Segundo a Wikipedia, o algoritmo de Euclides para o cálculo do *máximo divisor comum* pode ser escrito em pseudo-código como se segue:

```
function gcd(a, b)
    while a ≠ b
        if (a > b) a := a − b;
        else b := b − a;
    return a;
```

Escreva esta função em WhyML, incluindo a sua especificação. Note que poderá utilizar na especificação a função da biblioteca do Why3 `function gcd int int : int`.

```
module EuclideanAlgorithm

  use import int.Int
  use import int.EuclideanDivision
  use import ref.Refint
  use import number.Gcd

```

```
  let euclid (u v: int) : int
    requires { ... }
    ensures  { ... }
  =
    ...

```

Nota: se preferir apresentar uma implementação recursiva poderá fazê-lo, devendo nesse caso alterar `let` para `let rec`.

Exercício 2

Considere o seguinte tipo polimórfico de árvores binárias:

```
type tree 'a = Empty | Node (tree 'a) 'a (tree 'a)
```

Usando predicados referidos no exercício sobre árvores, escreva um contrato para a seguinte função que converte árvores ordenadas em listas ordenadas, preservando o número de ocorrências dos elementos.

```
let function tree_to_list  (t:tree int) : list int
```

Exercício 3

Complete a definição da função seguinte, que conta o número de ocorrências de um elemento `x` num array `u`, entre os índices `k` e `l-1`. Note que não é possível descrever logicamente, através de uma pós-condição, a contagem feita pela função; na verdade a ideia desta definição pura é ser utilizada ao nível lógico no exercício seguinte, para especificar o comportamento de uma outra.

```
let rec function numof (u:array int) (x:int) (k:int) (l:int)
  requires { ... }
  ensures { 0<=result<=l-k }
```

```
  variant { ... }
= ...

```

Exercício 4

O programa imperativo em baixo determina o elemento que ocorre com maior frequência num array ordenado. Trata-se de um problema nada trivial no caso geral, mas que no caso de o array estar ordenado tem uma solução de tempo linear óbvia.

Escreva o contrato correspondente a esta especificação informal e todas as restantes anotações necessárias para provar a correção do programa face a esse contrato. Deverá utilizar para isso a função `numof` do ex. anterior.

```
use int.Int
use ref.Refint
use array.Array
use array.NumOfEq

let most_frequent (a: array int) : int
= let ref r = a[0] in
  let ref c = 1 in
  let ref m = 1 in
  for i = 1 to length a - 1 do
    if a[i] = a[i-1] then begin
      incr c;
      if c > m then begin m <- c; r <- a[i] end
    end else
      c <- 1
  done;
  r
```