MATA77 Programação Funcional
Prova 1

Aluno:

Nenhuma das funções escritas nesta prova deve assumir o uso de registros de memórias modificáveis após a sua inicialização. Iterações devem ser realizadas somente com chamadas recursivas. Use funções somente se permitido.

Questão 1 (2,0 pontos)

Escreva em Racket a função `(gera-sequencia n1 n2 [f])` que gera uma lista com a sequência de números entre `n1` e `n2`, inclusive. Opcionalmente, está função recebe uma função `f` que é aplicada a cada número gerado. Por exemplo:

```
> (gera-sequencia 3 5)
 '(3 4 5)

> (gera-sequencia 3 5 (lambda(x) (* x x)))
 '(9 16 25)
```

Questão 2 (2 pontos) Escreva em Racket a função (**`empacote`** *`lista`*), que transforma uma lista em uma lista de listas, empacotando elementos iguais consecutivos em sublistas distintas. Por exemplo:

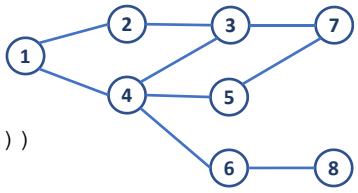
```
> (empacote '(a a a a a b b b b c c c a d e e e e e e))
'((a a a a a) (b b b b) (c c c) (a) (d) (e e e e e e))
```

Questão 3 (2,0 pontos)

Considere grafos <u>não direcionais</u> representados por uma lista de vértices e uma lista de pares de vértices (arestas), como exemplificado abaixo:

```
(define g1-lv '(1 2 3 4 5 6 7 8))

(define g1-la '((1 . 2) (1 . 4) (2 . 3) (4 . 3)
                (4 . 5) (4 . 6) (3 . 7) (5 . 7) (6 . 8)))
```

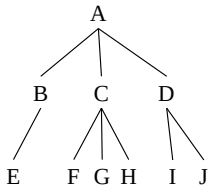


Escreva em Racket a função (**`adjacentes`** `v la`) que recebe um vértice V e a lista de aresta LA do grafo, e retorna a lista de vértices adjacentes a V no grafo. Por exemplo:

```
> (adjacentes 4 g1-la)
'(1 3 5 6)
```

Questão 4 (2,0 pontos)

Pode-se representar uma árvore genérica usando-se uma lista genérica da seguinte forma: `(define arv '(A (B (E)) (C (F) (G) (H)) (D (I) (J))))`, veja ao lado.



Escreva um programa em Racket que recebe uma árvore genérica representada no formato dado e retorna uma lista de pares com a profundidade de todos os seus nós. Exemplo:

```
> (prof-ag arv)
'((A . 1)(B . 2)(E . 3)(C . 2)(F . 3)(G . 3)(H . 3)(D . 2)(I . 3)(J . 3))
```

A ordem dos pares não importa.

Dica: se quiser você pode usar a função `length`, tamanho de uma lista, e `append`, concatenação de listas, para implementar a função pedida.

Questão 5 (2,0 pontos)

Escreva em Racket a função *`(dobra-d proc init lst ...+)`* que tem exatamente a mesma semântica de *`foldr`* da biblioteca do Racket. Por exemplo:

```
> (dobra-d + 0 '(1 2 3 4))
10

> (dobra-d cons '() '(1 2 3 4))
'(1 2 3 4)

> (dobra-d (lambda(x1 x2 x3 a)
             (cons x1 (cons x2 (cons x3 a))))
           '()
           '(1 2 3)
           '(a b c)
           '(10 20 30))
'(1 a 10 2 b 20 3 c 30)
```

Dica: você pode utilizar as funções `map` e `apply`.