Disciplina: Paradigmas de Programação
Professor: Maicon Rafael Zatelli
Entrega: *Moodle*

Trabalho III - Programação Funcional - LISP

Atenção: Faça um ZIP com todos os arquivos de solução. Use o nome do arquivo de maneira a entender qual problema você está resolvendo. Por exemplo, problema1.lisp, problema2.lisp e assim por diante. Responda todas as questões referentes ao problema 1, no arquivo do problema 1. Faça o mesmo para as questões referentes ao problema 2, e assim por diante. Para cada problema, inclua chamadas para suas funções, assim demonstrando o seu funcionamento.

- A legibilidade, organização do código, bem como o uso de comentários também serão considerados na avaliação.
- Note que a somatória dos pontos passa de 10, porém mesmo acertanto todos os problemas, a nota máxima será 10.
- **Crie um arquivo `nomes.txt` com o nome e matrícula dos membros da dupla.** Se vocês estiver fazendo individual, não crie este arquivo.

1. **(1.0)** Crie uma função **recursiva** `sequencia(n)` que retorne o n-ésimo número da sequência 1000, 990, 970, 940, 900, 850, ... (considere que 1000 é o elemento inicial da sequência).

2. **(1.0)** Crie uma função `palindrome(a)`, a qual recebe uma lista de números inteiros como parâmetro e deve retornar se a lista é uma palíndrome ou não. Uma lista é uma palíndrome se lendo os elementos nela contidos da esquerda para a direita e da direita para a esquerda forma-se a mesma sequência. Por exemplo, (1 2 3 4 4 3 2 1) é uma palíndrome, mas (1 2 3 1 4 3 2 1) não é uma palíndrome. Não utilize funções prontas do LISP para esta questão.

3. Motifique o arquivo `arvore.lisp` (disponível no Moodle) de forma a adicionar novas operações a nossa árvore. Assuma que nossa árvore não é uma árvore binária de busca.

   A (1.0): Crie uma função com a seguinte assinatura: `altura (arv)`, a qual recebe uma árvore como parâmetro e deve retornar a sua altura. Uma árvore com apenas o nó raiz possui altura 0. Assuma que não há nenhuma árvore com nenhum nó.

   B (1.0): Crie uma função com a seguinte assinatura: `igual (arvA arvB)`, a qual recebe duas árvores como parâmetro e deve retornar se elas são iguais. Duas árvores são iguais se elas possuem os mesmos elementos, dispostos da mesma forma.

   C (1.5): Crie uma função com a seguinte assinatura: `folhas (arv)`, a qual recebe uma árvore como parâmetro e deve retornar uma **lista** com todos os elementos que encontram-se nas folhas da árvore. Um nó é uma folha se ele não possui filhos.

   D (1.5): Crie uma função com a seguinte assinatura: `primoToNaoPrimo (arv)`, a qual recebe uma árvore como parâmetro e deve percorrer toda a árvore em busca dos números primos nela contidos. Ao encontrar um número primo, deve-se alterar o número daquele nó somando-se 1, para convertê-lo em um número não primo. Se ao somar 1 o número continuar primo, então deve-se somar 2, e assim por diante. Por exemplo, uma árvore contendo os números (7 4 5 6 15) deve resultar em uma árvore contendo os números (8 4 6 6 15).

4. **(1.5)** Crie uma função `repetidos(a m n)`, que recebe uma matriz de números inteiros com m linhas e n colunas e retorne uma lista contendo todos os elementos repetidos. Cada elemento deve aparecer uma única vez na lista de retorno, mesmo que ele apareça mais de 2 vezes na matriz.

5. **(2.5)** Para esta questão, considere um cenário de um labirinto, que pode ser representado por uma matriz de m linhas e n colunas e deseja-se saber se partindo-se da entrada, sempre por meio de alguma posição na coluna 0 pode-se chegar à saída, sempre por meio de alguma posição na coluna n-1. As paredes do labirinto são representadas pelo número 1 enquanto que as posições sem parede são representadas pelo número 0. Considere que somente é possível mover-se para a posição imediatamente superior ou imediatamente inferior ou à direita ou à esquerda da posição atual, ou seja, não pode-se mover pela diagonal e também não pode-se mover para posições onde existe parede, nem mesmo atravessar paredes. Por exemplo, no labirinto abaixo é possível encontrar uma saída por ambas as entradas.

```
1111111111111111
0001000011000001
1101111000011101
1101111111011101
1100000111011101
1110110111011101
1110001000010001
0010101111110111
1000000001110000
1111111111111111
```

Crie uma função `caminhoSaida(a m n)`, que recebe uma matriz de números inteiros com m linhas e n colunas, que representa o labirinto, e retorna o caminho partindo de alguma entrada até alguma saída saída, caso houver. O caminho deve ser ilustrado contendo o número 2 em cada posição da matriz por onde ele passa. Caso não houver caminho da entrada até a saída, retorne uma matriz com todas as posições contendo o número 3. Note que as entradas sempre estão na coluna 0 enquanto que as saídas sempre estão na coluna n-1. O exemplo de saída abaixo destaca o caminho de saída para a matriz acima.

```
1111111111111111
2221000011222221
1121111000211121
1121111111211121
1122222111211121
1110111211211121
1110001222212221
0010101111112111
1000000001112222
1111111111111111
```