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

Simulado - LISP

1. Crie uma função `sequencia(n)` que retorne o n-ésimo número da sequência 5, 11, 19, 29, 41, 55, ... (considere que 5 é o elemento inicial da sequência).

2. Crie uma função `pell(n)` que retorne o n-ésimo número de Pell. O n-ésimo número de Pell pode ser descoberto utilizando a seguinte fórmula:

$$P_n = \begin{cases} 0 & se\ n = 0 \\ 1 & se\ n = 1 \\ 2P_{n-1} + P_{n-2} & caso\ contrario \end{cases}$$

3. Crie uma função `interssecao(a b)`, a qual recebe duas listas de números inteiros (A e B) como parâmetro e deve retornar a intersseção entre elas, ou seja, uma terceira lista C com somente os elementos que estão em A e B. A lista C deve conter uma única vez cada número.

4. Crie uma função `segundoMenor(v)` que receba uma lista como parâmetro e retorne o segundo menor elemento nele.

5. Crie uma função `pico(a)` que receba uma matriz de m linhas e n colunas e retorne um pico na matriz. Um pico em uma matriz ocorre quando os elementos imediatamente à esquerda, à direita, acima e abaixo de um elemento são menores ou iguais a ele. Neste caso (observando a matriz abaixo) considere que x é um pico apenas se $x \geq a$ e $x \geq b$ e $x \geq c$ e $x \geq d$. Se x não tiver algum vizinho (devido a situar-se nas bordas da matriz), considere apenas os vizinhos existentes.

$$\begin{bmatrix} & a & \\ d & x & b \\ & c & \\ & & \end{bmatrix}$$

6. Crie uma função chamada `rotacionar(a, n)`, que receba uma matriz como parâmetro e rotacione a matriz para a direita n vezes, sendo n também recebido como parâmetro. O resultado da função deve ser a matriz rotacionada n vezes para a direita. Por exemplo, para a entrada abaixo e 2 rotações deve-se obter o resultado abaixo.

Entrada:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

Resultado:

$$\begin{bmatrix} 2 & 3 & 1 \\ 5 & 6 & 4 \\ 8 & 9 & 7 \\ 11 & 12 & 10 \end{bmatrix}$$

7. Dizemos que uma matriz quadrada inteira A_{nxn} é um quadrado mágico se a soma dos elementos de uma determinada linha, coluna ou diagonal é sempre igual. Faça que receba como parâmetro uma matriz com alguns números do quadrado mágico já preenchidos e retorne uma matriz com o quadrado mágico completo. Considere que números vão de 1 até 1000 (inclusive) e podem se repetir. As posições da matriz com 0 indicam que aquela posição não está preenchida. Abaixo, são ilustrados dois exemplos de matrizes dadas como entrada e o resultado esperado da sua função. Note que podem existir vários resultados válidos, mas também pode ocorrer de não existir uma solução para a matriz dada. Neste caso, retorne uma matriz toda zerada.

Entrada:

$$\begin{bmatrix} 0 & 12 & 12 \\ 16 & 10 & 0 \\ 8 & 0 & 0 \end{bmatrix}$$

Resultado:

$$\begin{bmatrix} 6 & 12 & 12 \\ 16 & 10 & 4 \\ 8 & 8 & 14 \end{bmatrix}$$

Entrada:

$$\begin{bmatrix} 0 & 468 & 0 \\ 0 & 522 & 414 \\ 441 & 0 & 549 \end{bmatrix}$$

Resultado:

$$\begin{bmatrix} 495 & 468 & 603 \\ 630 & 522 & 414 \\ 441 & 576 & 549 \end{bmatrix}$$

8. Motifique o arquivo `arvore.hs` (disponível no Moodle, na atividade sobre estruturas em LISP) de forma a adicionar novas operações a nossa árvore:

   A: Crie uma função com a seguinte assinatura: `maioresQueElemento(arv, x)`, a qual recebe um número e deve retornar a quantidade de números maiores que ele na árvore.

   B: Crie uma função com a seguinte assinatura: `pares(arv)`, a qual deve retornar uma lista contendo todos os números pares da árvore.

   C: Crie uma função com a seguinte assinatura: `folhas(arv)`, a qual deve retornar a quantidade de folhas na árvore. Um nó é uma folha se ele não possui filhos.

   D: Crie uma função com a seguinte assinatura: `maiorFinonacci(arv)`, a qual deve retornar o maior número da sequência de Fibonacci encontrado na árvore. Caso não haja nenhum, retorne -1. O n-ésimo número de Fibonacci pode ser descoberto utilizando a seguinte fórmula:

$$F_n = \begin{cases} 0 & se\ n = 0 \\ 1 & se\ n = 1 \\ F_{n-1} + F_{n-2} & caso\ contrario \end{cases}$$

   E: Crie uma função com a seguinte assinatura: `inserir(arv, x)`, a qual deve inserir um novo elemento em um ramo da árvore de menor distância da folha até a raiz. DICA: utilize as funções **setf** e **make**.