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

Atividade II - 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.

Resolva os seguintes problemas na linguagem LISP:

1. Crie uma função com assinatura `soma`, a qual recebe uma lista de inteiros e retorna a soma de todos os elementos da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
2. Crie uma função com assinatura `media`, a qual recebe uma lista de inteiros e retorna a média de todos os elementos da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
3. Crie uma função com assinatura `menor`, a qual recebe uma lista de inteiros e retorna o menor elemento da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
4. Crie uma função com assinatura `diferencaMaiorMenor`, a qual recebe uma lista de inteiros e retorna a diferença entre o maior e o menor elemento da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
5. Crie uma função com assinatura `busca`, a qual recebe uma lista de inteiros e um inteiro e retorna se o elemento passado como parâmetro encontra-se na lista ou não. **Não utilize** nenhuma função pronta do LISP para realizar esta tarefa.
6. Crie uma função com assinatura `ocorrencias`, a qual recebe uma lista de inteiros e um inteiro e retorna o número de vezes em que o elemento está presente na lista. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
7. Crie uma função com assinatura `enesimo`, a qual recebe uma lista de inteiros e um inteiro n e retorna o n-ésimo elemento na lista. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
8. Crie uma função com assinatura `fatia`, a qual recebe uma lista de inteiros, um inteiro n e um inteiro m e retorna todos os elementos da lista a partir da posição n (inclusive) até a posição m (exceto o elemento da posição m). **Não utilize** nenhuma função pronta to LISP para esta tarefa.
9. Crie uma função com assinatura `inverte`, a qual recebe uma lista como parâmetro e deve retornar a mesma invertida. **Não utilize** nenhuma função pronta do LISP para realizar esta tarefa.
10. Crie uma função com assinatura `mapear`, a qual recebe uma função e uma lista e retorna uma lista. Esta função `mapear` fará o mesmo que a função `map` do Haskell, ou seja, aplicar a função recebida como parâmetro sobre cada elemento da lista e retornar a lista resultante. **Não utilize** nenhuma função pronta do LISP para realizar esta tarefa.
11. Crie uma função com assinatura `primeiros`, a qual recebe um número de elementos, uma lista, e retorna uma lista. Esta função deve retornar uma lista com os n primeiros elementos informados no primeiro parâmetro. **Não utilize** nenhuma função pronta to LISP para esta tarefa.
12. Crie uma função com assinatura `apagar`, a qual recebe, um número de elementos (n), uma lista, e retorna uma lista. Esta função deve remover da lista os n primeiros elementos fornecidos como parâmetro. Por exemplo, a chamada `(apagar 3 '(1 2 3 4 5))` deve retornar `(4 5)`. **Não utilize** nenhuma função pronta to LISP para esta tarefa.

13. Crie uma função com assinatura `apagarEnquanto`, a qual recebe uma função como parâmetro e uma lista, e retorna uma lista. Esta função deve aplicar a função passada como parâmetro a cada elemento da lista, até que algum elemento da lista retorne `NIL` na aplicação da função. Os elementos da lista resultante são então todos os elementos a partir do elemento em que a função passada como parâmetro retornou `NIL`. Por exemplo a chamada `(apagarEnquanto par '(2 4 1 2 3 4 5))` deve retornar `(1 2 3 4 5)`, visto que ao testar o elemento 1, a função `par` retorna `NIL`. **Não utilize** nenhuma função pronta to LISP para esta tarefa.

14. Motifique o arquivo `alunos.lisp` (disponível no Moodle) de forma a adicionar novas funções:

    A: Crie uma função com o seguinte nome: `medias`, a qual recebe uma lista de alunos e retorna uma lista de duplas, onde cada dupla contém o nome e a média de cada aluno. Note que cada aluno pode ter um número diferente de notas.

    B: Crie uma função com o seguinte nome: `mediaTurma`, a qual recebe uma lista de alunos e retorna a média da turma.

    C: Crie uma função com o seguinte nome: `acimaMedia`, a qual recebe uma lista de alunos e retorna os nomes e médias dos alunos que estão acima da média da turma.

    D: Crie uma função com o seguinte nome: `aprovados`, a qual recebe uma lista de alunos e retorna uma lista com o nome dos alunos aprovados. Um aluno está aprovado se a sua média é maior ou igual a 6.

    E: Crie uma função com o seguinte nome: `duplas`, a qual recebe uma lista de alunos e retorna uma lista de duplas de alunos. Note que um aluno não pode fazer dupla consigo mesmo.

    F: Crie uma função com o seguinte nome: `ordernarAlunos`, a qual recebe uma lista de alunos e retorna os nomes e médias dos alunos ordenados em ordem crescente de média.