



UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
Rua Dom Manoel de Medeiros, s/n – Dois Irmãos 52171-900 Recife-PE
Fone: 81 3302 1000 www.dc.ufrpe.br

DISCIPLINA: Algoritmos e Estruturas de Dados	CÓDIGO: 06214
DEPARTAMENTO: Computação	ÁREA: Informática
CURSO: Licenciatura em Computação	
PROFESSOR RESPONSÁVEL: Luciano Demétrio Santos Pacífico	
DATA MÁXIMA DE ENTREGA: 26-09-2020	

Regras da Lista de Exercícios 05

1. **Não é permitido o uso de Estruturas de Dados prontas de Linguagens de Programação.** O aluno deve implementar suas próprias Estruturas de Dados. Na Linguagem de Programação **C**, deve-se usar **structs**. Nas demais Linguagens de Programação permitidas (vide **Regras da Disciplina**), deve-se usar **classes**.
2. **Não é permitido o uso de Algoritmos e comandos otimizados prontos de Linguagens de Programação.** Todos os algoritmos solicitados devem ser implementados pelos alunos como **procedimentos** (funções, métodos, etc.).
3. As questões que solicitam **escrita de código** devem ser resolvidas **apenas através dos recursos oferecidos pela pseudolingagem definida para a disciplina, e dos recursos equivalentes em Linguagens de Programação reais**, sendo eles: variáveis, constantes e tipos primitivos, expressões, estruturas condicionais, estruturas de repetição, sub-rotinas, estruturas de dados homogêneas (**Arrays**) e estruturas de dados heterogêneas (registros – classes e structs).
4. Para a Lista de Exercícios 05, os **Arrays** devem ser implementados através de **Alocação Estática**. Não será permitido o uso de **Estruturas de Dados dinâmicas** implementadas em Linguagens de Programação, como os **Vectors** e **Lists** da Linguagem de Programação **Java**, por exemplo. Deve-se usar **arrays**.
5. Como a Linguagem de Programação **Python** não suporta **arrays estáticos**, o aluno que optar por usar esta Linguagem deverá usar **Lists**, **única e exclusivamente para simular o comportamento de arrays estáticos**, de acordo com os seguintes critérios:
 - a. Deve-se criar uma **List** com posições vazias através do método **append** dessa estrutura. O uso do **append** será permitido **apenas na alocação de memória para a variável que representará o array estático**;
 - b. O limite máximo de **M** objetos deve ser controlado **através de código**;
 - c. A Estrutura deve ser manipulada como se fosse um **array estático**, com os **procedimentos escritos pelo aluno, não sendo permitido** o uso de métodos, funções ou otimizações oferecidos pela Linguagem Python.
6. **Regra de Ouro:** Todos os alunos envolvidos em **cópias** terão suas notas **ANULADAS** nas referidas questões.
7. Apenas o código “.c”, “.cpp”, “.java”, “.py”, etc. deve ser enviado ao professor para cada questão. Deve-se enviar **um único arquivo resposta por questão**, que conterá todas as classes/estruturas e procedimentos necessários para a solução da questão. Todos os arquivos devem ser enviados **em uma única pasta, “zipados”**.

8. O arquivo de resposta com o código para cada questão deve ser nomeado na forma “L#Q%.c”, “L#Q%.java”, etc., onde “#” refere-se ao número da lista e “%” refere-se ao número da questão (ex.: L1Q2.c, para o arquivo de resposta da segunda questão da primeira lista, usando a Linguagem C).
9. A resposta do Lista de Exercício 05 deve ser submetida **unicamente através da tarefa criada no Google Classroom para este propósito**.

Lista de Exercícios 05 – Tabelas *Hash*

1. Escreva, usando uma **Linguagem de Programação**, o algoritmo de **inserção** em uma Tabela *Hash* de tamanho m (parâmetro fornecido como entrada para o algoritmo), que executa a estratégia de ***Open Address Hashing por Linear Probing*** enquanto houver espaço disponível na tabela. Quando a tabela estiver completamente preenchida, o algoritmo deve executar a estratégia de ***Closed Address Hashing***. Escreva o código para as **Estruturas de Dados** necessárias. Considere a *hash function* abaixo para uma chave k qualquer:

$$h(k) = k \bmod m$$

Obs.: A Tabela *Hash* propriamente dita deve ser implementada como um **Array estático**, enquanto as listas usadas pela estratégia de ***Closed Address Hashing***, devem ser implementadas como **Listas Lineares Encadeadas** (alocação dinâmica). (10.0 pontos)