**ANO**

**2025**



**ATIVIDADE PRÁTICA**

ESTRUTURA DE DADOS /

PROGRAMAÇÃO III

**ROBSON AUGUSTO DOS SANTOS. RU: 3621366**

**Prof. Me. Bruno Kostiuk**

QUESTÃO 1 de 2 – Lista Encadeada

**Enunciado:** Com a finalidade de melhorar o atendimento e priorizar os casos mais urgentes, a direção de um hospital criou um sistema de triagem em que um profissional da saúde classifica a ordem de atendimento com base numa avaliação prévia do paciente, entregando-lhe um cartão numerado verde (V) ou amarelo (A), que define o menor ou maior grau de urgência da ocorrência, respectivamente. Para informatizar esse processo, a direção do hospital contratou você para desenvolver uma fila de chamada seguindo as seguintes regras:

* Pacientes com cartão numerado amarelo (A) são chamados antes dos pacientes com cartão numerado verde (V)
* Entre os pacientes com cartão numerado amarelo (A), os que tem numeração menor são atendidos antes.
* Entre os pacientes com cartão numerado verde (V), os que tem numeração menor são atendidos antes.
* As numerações dos cartões verdes (V) iniciam em 1.
* As numerações dos cartões amarelos (A) iniciam em 201.

Elabore um programa em Python que:

1. Deve-se implementar uma **Lista Encadeada Simples** em que**:** [EXIGÊNCIA DE CÓDIGO 1 de 7];
   1. O Nodo representa um cartão numerado contendo: **número**, **cor** e um ponteiro para o **próximo**;
   2. A lista é não circular, ou seja, seu último elemento aponta para nulo;
2. Deve-se implementar a função **inserirSemPrioridade(nodo)** em que**:** [EXIGÊNCIA DE CÓDIGO 2 de 7];
   1. Deve-se andar pela lista a partir da cabeça (***head***) e inserir o nodo no **final da lista**.
3. Deve-se implementar a função **inserirComPrioridade(nodo)** em que**:** [EXIGÊNCIA DE CÓDIGO 3 de 7];
   1. Deve-se andar pela lista a partir da cabeça (***head***) e inserir o nodo **após todos os nodos com cor “A” que estão na lista**.
   2. O nodo inserido deve **sempre** estar antes de todos os nodos com cor “V”.
4. Deve-se implementar a função **inserir()** em que**:** [EXIGÊNCIA DE CÓDIGO 4 de 7];
   1. Deve-se solicitar ao usuário a cor (“A” ou “V”).
   2. A partir da cor, o número (inteiro) do paciente deve ser **atribuído automaticamente** seguindo a ordem numérica. Por exemplo: o primeiro paciente “V” será o 1, o segundo 2, e assim por diante.
   3. Deve-se criar um nodo com a cor e o número atribuído ao paciente.
   4. Se a lista estiver vazia, a cabeça (***head***) da lista deve apontar para o nodo criado. Senão, se a cor do nodo for “V”, deve-se chamar a função **inserirSemPrioridade(nodo).** Senão, se a cor do nodo for “A”, deve-se chamar a função **inserirComPriordade(nodo).**
5. Deve-se implementar a função **imprimirListaEspera()** em que: [EXIGÊNCIA DE CÓDIGO 5 de 7];
   1. Deve-se imprimir todos os cartões e seus respectivos números a partir do primeiro até o último da lista.
6. Deve-se implementar a função **atenderPaciente()** em que: [EXIGÊNCIA DE CÓDIGO 6 de 7];
   1. Deve-se remover o primeiro paciente da fila e imprimir uma mensagem chamando o paciente para atendimento informando o número do seu cartão.
7. Deve-se implementar um menu para utilização do sistema em que: [EXIGÊNCIA DE CÓDIGO 7 de 7];
   1. Deve-se apresentar as opções (1 – adicionar paciente a fila, 2 – mostrar pacientes na fila, 3 – chamar paciente, 4 – sair)
   2. Se escolhida a opção 1, chamar a função **inserir().**
   3. Se escolhida a opção 2, chamar a função **imprimirListaEspera().**
   4. Se escolhida a opção 3, chamar a função **atenderPaciente().**
   5. Se escolhida a opção 4, encerrar o programa.
   6. Se escolhida uma opção diferente as opções disponíveis, volte para o menu.

Para testar o software, execute os seguintes passos e apresente a saída do console conforme exemplo de saída de console (próxima página):

1. Deve-se testar o sistema inserindo três (3) pacientes com cartão de cor “V”, dois (2) pacientes com cartão de cor “A”, dois (2) pacientes com cartão “V” e três (3) pacientes com cartão de cor “A”, **nessa respectiva ordem**. [EXIGÊNCIA DE SAÍDA DE CONSOLE 1 de 3];
2. Deve-se apresentar na saída de console a impressão da lista de espera (opção 2 do menu principal). [EXIGÊNCIA DE SAÍDA DE CONSOLE 2 de 3];
3. Deve-se apresentar na saída de console o atendimento de dois (2) pacientes (opção 3 do menu principal) e em seguida mostrar a lista de espera (opção 2 do menu principal). [EXIGÊNCIA DE SAÍDA DE CONSOLE 3 de 3];

Apresentação de **Código da Questão 1**:

# Função principal [EXIGÊNCIA DE CÓDIGO 1 de 7];

def main():

head = None

while True:

# [EXIGÊNCIA DE CÓDIGO 7 de 7];

print("\n1 - Adicionar paciente a fila")

print("2 - Mostrar pacientes na fila")

print("3 - Chamar paciente")

print("4 - Sair")

selecao = int(input("Escolha uma opção acima: >> "))

if selecao == 1:

head = inserir(head)

elif selecao == 2:

imprimirListaEspera(head)

elif selecao == 3:

head = atenderPaciente(head)

elif selecao == 4:

break

class Nodo:

def \_\_init\_\_(self, numero, cor):

self.numero = numero

self.cor = cor

self.proximo = None

# Função para inserir no final da lista [EXIGÊNCIA DE CÓDIGO 2 de 7];

def inserirSemPrioridade(Nodo, head):

if not head:

head = Nodo

else:

atual = head

while atual.proximo:

atual = atual.proximo

atual.proximo = Nodo

return head

# Função para inserir após todos os Nodos com cor "A" [EXIGÊNCIA DE CÓDIGO 3 de 7];

def inserirComPrioridade(Nodo, head):

if not head:

head = Nodo

else:

if Nodo.cor == "A":

if Nodo.numero <= 201:

Nodo.proximo = head

head = Nodo

else:

atual = head

while atual.proximo and atual.proximo.cor == "A" and atual.proximo.numero <= 201:

atual = atual.proximo

Nodo.proximo = atual.proximo

atual.proximo = Nodo

else:

atual = head

while atual.proximo and atual.proximo.cor == "A":

atual = atual.proximo

Nodo.proximo = atual.proximo

atual.proximo = Nodo

return head

# Função para inserir um paciente na fila [EXIGÊNCIA DE CÓDIGO 4 de 7];

def inserir(head):

cor = input("Digite a cor do cartão (A/V): ")

if head == None:

if cor == "V":

numero = 1

else:

numero = 201

else:

paciente = None

atual = head

while atual:

if atual.cor == cor and (paciente is None or atual.numero > paciente.numero):

paciente = atual

atual = atual.proximo

if paciente == None and cor == "V":

numero = 1

elif paciente == None and cor == "A":

numero = 201

else:

numero = paciente.numero+1

Nodo\_Nodo = Nodo(numero, cor)

if cor == "V":

head = inserirSemPrioridade(Nodo\_Nodo, head)

elif cor == "A":

head = inserirComPrioridade(Nodo\_Nodo, head)

return head

# Função para imprimir todos os cartões e seus respectivos números a partir do primeiro até o último da lista [EXIGÊNCIA DE CÓDIGO 5 de 7];

def imprimirListaEspera(head):

pacientes = []

atual = head

while atual:

pacientes.append(atual)

atual = atual.proximo

pacientes.sort(key=lambda paciente: (paciente.cor != "A", paciente.numero))

for paciente in pacientes:

print(f"Cartão {paciente.cor}: {paciente.numero}")

# Função para atender o próximo paciente [EXIGÊNCIA DE CÓDIGO 6 de 7];

def atenderPaciente(head):

if not head:

print("Sem pacientes na fila.")

return head

paciente\_atender = None

atual = head

while atual:

if atual.cor == "A" and (paciente\_atender is None or atual.numero < paciente\_atender.numero):

paciente\_atender = atual

atual = atual.proximo

# Se não tiver pacientes com cartão A, atender o primeiro da fila

if paciente\_atender is None:

paciente = head

head = head.proximo

else:

# Remover o paciente a ser atendido da lista

if paciente\_atender == head:

head = head.proximo

else:

anterior = head

while anterior.proximo != paciente\_atender:

anterior = anterior.proximo

anterior.proximo = paciente\_atender.proximo

paciente = paciente\_atender

print(f"Chamando paciente com cartão {paciente.cor}: {paciente.numero}")

return head

main()

Apresentação de **Saída do Console da Questão 1**:

Texto

O conteúdo gerado por IA pode estar incorreto. Texto

O conteúdo gerado por IA pode estar incorreto.

**Figura 1: Exemplo de saída de console que o aluno deve fazer. Em que se insere 10 pacientes (5 com cartão verde e 5 com cartão amarelo) conforme** [EXIGÊNCIA DE SAÍDA DE CONSOLE 1 de 3];

Texto

O conteúdo gerado por IA pode estar incorreto.

**Figura 2: Exemplo de saída de console que o aluno deve fazer. Em que mostra a lista de pacientes, conforme** [EXIGÊNCIA DE SAÍDA DE CONSOLE 2 de 3];

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Figura 3: Exemplo de saída de console que o aluno deve fazer. Em que ele chama dois pacientes para atendimento e em seguida mostra a lista de pacientes, conforme** [EXIGÊNCIA DE SAÍDA DE CONSOLE 3 de 3];

QUESTÃO 2 de 2 – Tabela Hash

**Enunciado:** Com o objetivo de criar um sistema novo de emplacamento de veículos, deputados em do Distrito Federal – DF, decidiram que o último número da placa dos veículos, irá representar o estado de registro dele. Para isso, sua equipe de desenvolvedores foi encarregada de desenvolver uma **Tabela Hash com endereçamento em cadeia de** **10 posições** (cada posição do vetor deve ser uma lista encadeada), representando os números de 0 a 9 que irão representar os 26 estados e o Distrito Federal (total 27).

A função hash deve seguir as seguintes regras:

* A entrada da função hash deve ser uma string com 2 letras, representando a sigla do estado e/ou distrito federal.
* Caso a sigla seja DF (Distrito Federal), por questões de superstição, os deputados solicitaram que o retorno da função seja 7 sempre.
* Caso contrário, a função deve retornar a posição com base no valor ASCII das duas letras e seguindo a seguinte regra:

Onde e são os valores ASCII da primeira e segunda letra, respectivamente (Tabela ASCII no final do documento).

Elabore um programa em Python que:

1. Deve-se implementar a tabela Hash com 10 posições, onde inicialmente todas as posições possuem valor **None** [EXIGÊNCIA DE CÓDIGO 1 de 7];
2. Deve-se implementar as **Listas Encadeadas Simples** em que**:** [EXIGÊNCIA DE CÓDIGO 2 de 7];
   1. O Nodo representa um Estado contendo: **sigla**, **nomeEstado** e um ponteiro para o **próximo**;
   2. As 10 posições da tabela hash, representam a cabeça de cada lista (***head***).
3. Deve-se implementar a inserção no início da lista encadeada (cada elemento novo deve ser sempre **inserido no início da lista**) [EXIGÊNCIA DE CÓDIGO 3 de 7];
4. Deve-se implementar a impressão da tabela hash, onde devem ser impressas as **siglas** de todos os nodos que estão na tabela hash **separados por posição** [EXIGÊNCIA DE CÓDIGO 4 de 7];
5. Deve-se implementar a função hash, conforme enunciado. [EXIGÊNCIA DE CÓDIGO 5 de 7];
6. Deve-se implementar a inserção dos estados e distrito federal (**todos os 27 com nome e sigla**) na tabela hash utilizando a função hash (não precisa solicitar ao usuário a digitação via teclado, pode inserir no código mesmo de modo *hard code*) [EXIGÊNCIA DE CÓDIGO 6 de 7];
7. Deve-se inserir na Tabela, além dos estados e distrito federal, um estado fictício, sendo que esse estado tenha seu **nome completo** e como siglas, a primeira letra do seu nome e a primeira letra do seu último sobrenome. Exemplo: Bruno Kostiuk – BK. EXIGÊNCIA DE CÓDIGO 7 de 7];

Para testar o software, execute os seguintes passos e apresente a saída do console conforme exemplo de saída de console (próxima página):

1. Deve-se apresentar na saída de console, a impressão da tabela hash antes de inserir qualquer informação [EXIGÊNCIA DE SAÍDA DE CONSOLE 1 de 3];
2. Deve-se apresentar na saída de console, a impressão da tabela hash após inserir os 26 estados e o Distrito Federal - DF [EXIGÊNCIA DE SAÍDA DE CONSOLE 2 de 3];
3. Deve-se apresentar na saída de console, a impressão da tabela hash após inserir os 26 estados, Distrito Federal – DF e o estado fictício com seu nome completo. [EXIGÊNCIA DE SAÍDA DE CONSOLE 3 de 3];

A imagem apresenta a tabela ASCII disponível também em:

https://www.matematica.pt/util/resumos/tabela-ascii.php

Apresentação de **Código da Questão 2**:

class Estado:

def \_\_init\_\_(self, sigla, nome):

self.sigla = sigla

self.nome = nome

self.proximo = None

class Hash: #[EXIGÊNCIA DE CÓDIGO 5 de 7];

def \_\_init\_\_(self):

self.tamanho = 10

self.tabela = [None] \* self.tamanho #[EXIGÊNCIA DE CÓDIGO 1 de 7]

def funcao\_hash(self, sigla):

if sigla == "DF":

return 7

else:

char1\_ascii = ord(sigla[0])

char2\_ascii = ord(sigla[1])

return (char1\_ascii + char2\_ascii) % self.tamanho

def inserir(self, estado): #[EXIGÊNCIA DE CÓDIGO 2 de 7];

posicao = self.funcao\_hash(estado.sigla)

if self.tabela[posicao] is None:

self.tabela[posicao] = estado

else: #[EXIGÊNCIA DE CÓDIGO 3 de 7];

estado.proximo = self.tabela[posicao]

self.tabela[posicao] = estado

def imprimir(self): #[EXIGÊNCIA DE CÓDIGO 4 de 7];

for i, estado in enumerate(self.tabela):

estados\_na\_posicao = []

while estado:

estados\_na\_posicao.append(estado.sigla)

estado = estado.proximo

estados\_na\_posicao.append("None")

print(f"{i}: {'->'.join(estados\_na\_posicao)}")

estados = [ #[EXIGÊNCIA DE CÓDIGO 6 de 7];

Estado("AC", "Acre"),

Estado("AL", "Alagoas"),

Estado("AP", "Amapa"),

Estado("AM", "Amazonas"),

Estado("BA", "Bahia"),

Estado("CE", "Ceara"),

Estado("ES", "Espirito Santo"),

Estado("GO", "Goias"),

Estado("MA", "Maranhão"),

Estado("MT", "Mato Grosso"),

Estado("MS", "Mato Grosso do Sul"),

Estado("MG", "Minas Gerais"),

Estado("PA", "Para"),

Estado("PB", "Paraiba"),

Estado("PR", "Parana"),

Estado("PE", "Pernambuco"),

Estado("PI", "Piaui"),

Estado("RJ", "Rio de Janeiro"),

Estado("RN", "Rio Grande do Norte"),

Estado("RS", "Rio Grande do Sul"),

Estado("RO", "Rondonia"),

Estado("RR", "Roraima"),

Estado("SC", "Santa Catarina"),

Estado("SP", "Sao Paulo"),

Estado("SE", "Sergipe"),

Estado("TO", "Tocantins"),

Estado("DF", "Distrito Federal") ]

tabela\_hash = Hash() #instanciando Hash vazio (None)

tabela\_hash.imprimir() #[EXIGÊNCIA DE SAÍDA DE CONSOLE 1 de 3];

print("\n")

for i in range(len(estados)):

tabela\_hash.inserir(estados[i]) #inserindo estados no hash. [EXIGÊNCIA DE CÓDIGO 7 de 7];

#imprimindo a tabela hash sem o estado fictício

tabela\_hash.imprimir() #[EXIGÊNCIA DE SAÍDA DE CONSOLE 2 de 3];

print("\n")

estado\_ficticio = Estado("RA", "Robson Augusto dos Santos") #declarando estado ficticio

tabela\_hash.inserir(estado\_ficticio) #inserindo estado ficticio

tabela\_hash.imprimir() #[EXIGÊNCIA DE SAÍDA DE CONSOLE 3 de 3];

Apresentação de **Saída do Console da Questão 2**:

**Texto

O conteúdo gerado por IA pode estar incorreto.**

Figura 1: Exemplo de saída de console que o aluno deve fazer. Impressão da tabela hash antes de inserir qualquer informação, conforme [EXIGÊNCIA DE SAÍDA DE CONSOLE 1 de 3];

**Texto

O conteúdo gerado por IA pode estar incorreto.**

Figura 2: Exemplo de saída de console que o aluno deve fazer. Impressão da tabela hash após inserir os 26 estados e o Distrito Federal - DF, conforme [EXIGÊNCIA DE SAÍDA DE CONSOLE 2 de 3];

**Texto

O conteúdo gerado por IA pode estar incorreto.**

Figura 3: Exemplo de saída de console que o aluno deve fazer. Impressão da tabela hash após inserir os 26 estados, Distrito Federal – DF e o estado fictício com seu nome completo (No caso foi inserido BK na posição 1), conforme [EXIGÊNCIA DE SAÍDA DE CONSOLE 3 de 3];