**ANO**

**2025**



**ATIVIDADE PRÁTICA**

ESTRUTURA DE DADOS /

PROGRAMAÇÃO III

**Carlos Henrique Monnerat Quintanilha. RU: 4328237**

**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**:

# Classe que representa uma paciente na fila  
class Paciente:  
 def \_\_init\_\_(self, numero, cor):  
 self.numero = numero # numero do cartão do paciente  
 self.cor = cor # cor do cartão (A(amarelo) ou V(verde))  
 self.prox = None # próximo paciente da fila  
  
# Classe da fila de pacientes (lista encadeada simples)  
class FilaHospital:  
 def \_\_init\_\_(self):  
 self.inicio = None # cabeca da lista  
 self.contadorV = 1 # começa numeração dos verdes em 1  
 self.contadorA = 201 # começa numeração dos amarelos em 201  
  
 # insere paciente no final da fila (sem prioridade)  
 def adicionarNoFinal(self, novo):  
 if self.inicio == None:  
 self.inicio = novo # se não tem ninguém, ele é o primeiro  
 else:  
 atual = self.inicio  
 while atual.prox != None:  
 atual = atual.prox  
 atual.prox = novo # coloca no final da fila  
  
 # insere paciente com prioridade (depois dos amarelos, antes dos verdes)  
 def adicionarComPrioridade(self, novo):  
 if self.inicio == None or self.inicio.cor == "V":  
 novo.prox = self.inicio  
 self.inicio = novo  
 else:  
 atual = self.inicio  
 while atual.prox != None and atual.prox.cor == "A":  
 atual = atual.prox  
 novo.prox = atual.prox  
 atual.prox = novo  
  
 # função para adicionar paciente, perguntando a cor e atribuindo número automaticamente  
 def adicionarPaciente(self):  
 cor = input("Digite a cor do cartão (A (amarelo) ou V (verde)): ").upper()  
 if cor == "V":  
 numero = self.contadorV  
 self.contadorV += 1  
 elif cor == "A":  
 numero = self.contadorA  
 self.contadorA += 1  
 else:  
 print("Cor inválida! Digite A ou V.")  
 return  
  
 novo = Paciente(numero, cor)  
  
 if self.inicio == None:  
 self.inicio = novo # se a fila estiver vazia  
 elif cor == "V":  
 self.adicionarNoFinal(novo)  
 elif cor == "A":  
 self.adicionarComPrioridade(novo)  
  
 print("Paciente adicionado com cartão", cor + str(numero))  
  
 # Mostra todos os pacientes da fila  
 def mostrarFila(self):  
 if self.inicio == None:  
 print("A fila está vazia!")  
 else:  
 atual = self.inicio  
 print("Fila de espera:")  
 while atual != None:  
 print("Cartão:", atual.cor + str(atual.numero))  
 atual = atual.prox  
  
 # chama o proximo paciente da fila (o primeiro)  
 def chamarPaciente(self):  
 if self.inicio == None:  
 print("Nenhum paciente na fila.")  
 else:  
 print("Chamando paciente com cartão", self.inicio.cor + str(self.inicio.numero))  
 self.inicio = self.inicio.prox # remove o primeiro da fila  
  
 # menu com opções para o usuário  
 def menu(self):  
 while True:  
 print("\n===== MENU =====")  
 print("1 - Adicionar paciente")  
 print("2 - Mostrar fila")  
 print("3 - Chamar paciente")  
 print("4 - Sair")  
 op = input("Escolha uma opção: ")  
  
 if op == "1":  
 self.adicionarPaciente()  
 elif op == "2":  
 self.mostrarFila()  
 elif op == "3":  
 self.chamarPaciente()  
 elif op == "4":  
 print("Saindo do sistema...")  
 break  
 else:  
 print("Opção errada, digite de novo!")  
  
# Parte que roda o sistema  
if \_\_name\_\_ == "\_\_main\_\_":  
 sistema = FilaHospital()  
 sistema.menu()

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

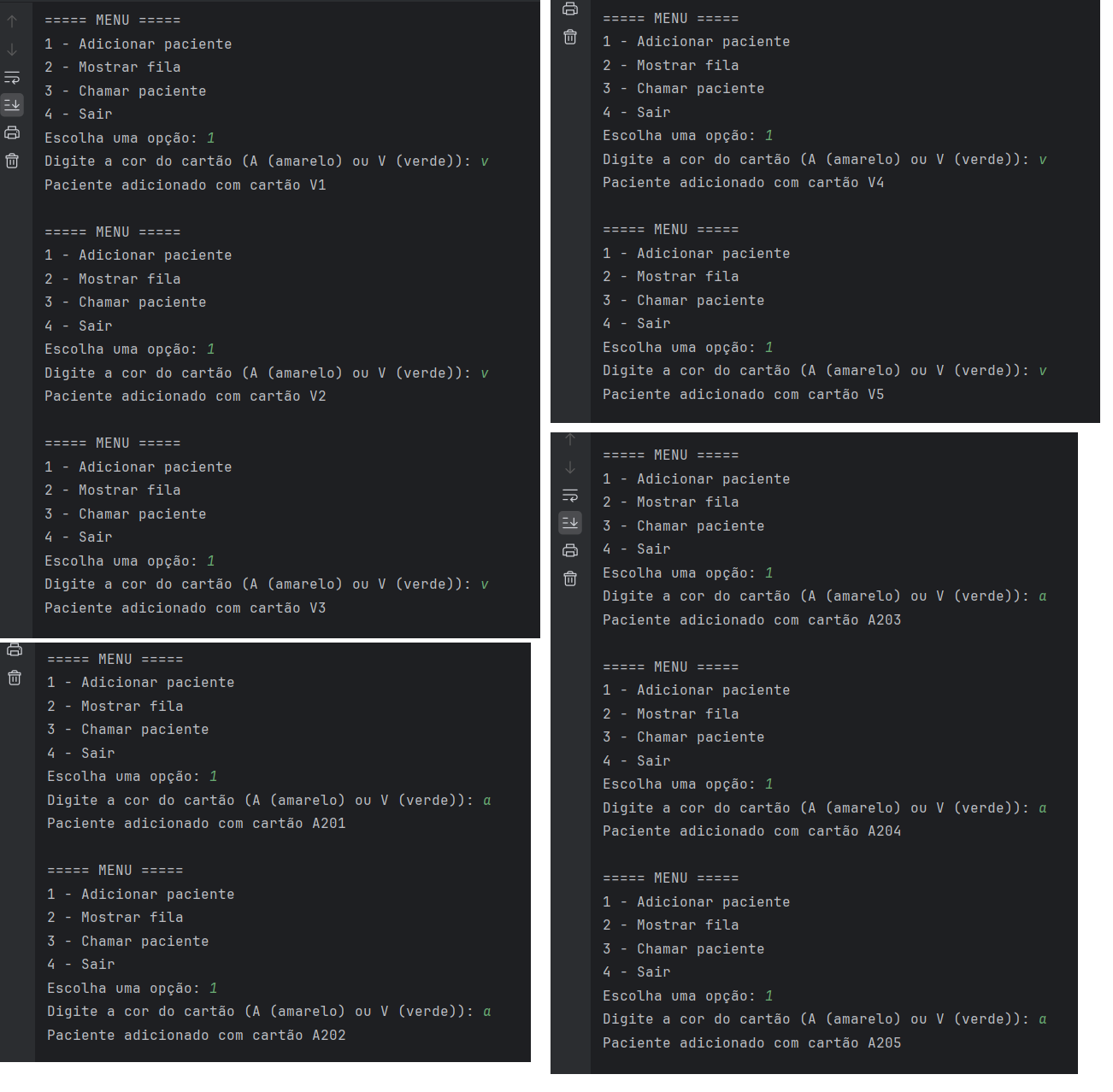
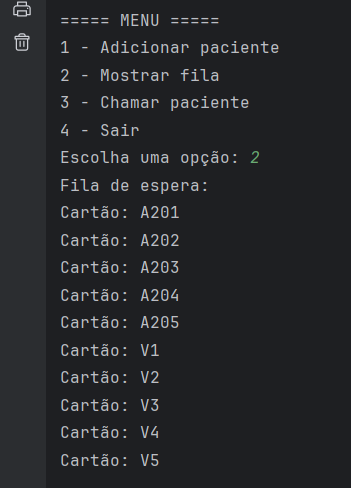
****

Figura 1: Saída do console em que se insere 10 pacientes (5 com cartão verde e 5 com cartão amarelo).

****

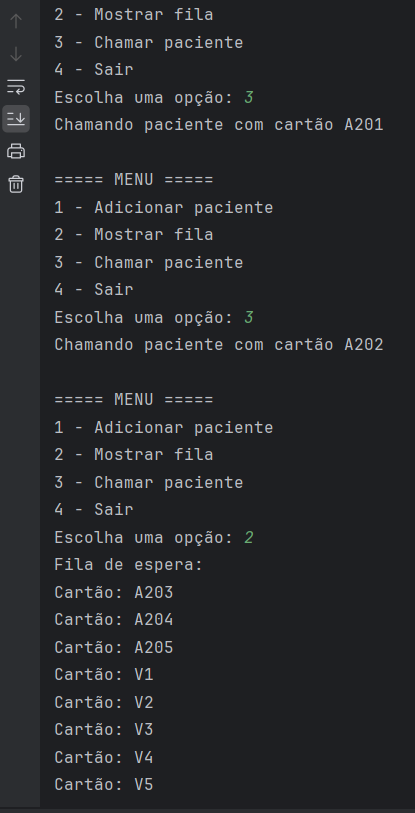
****

Figura 2: Saída de console em que mostra a lista de pacientes.

Figura 3: Saída do console em que chama dois pacientes para atendimento e em seguida mostra a lista de pacientes

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 de10 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];

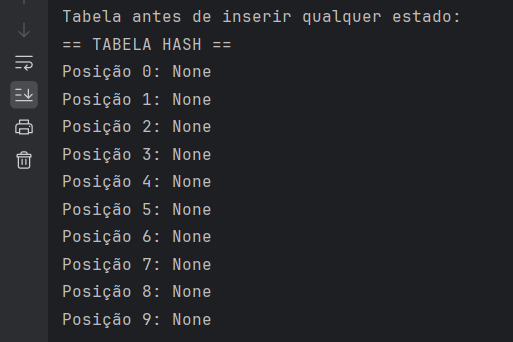
A imagem apresenta a tabela ASCII disponível também em:

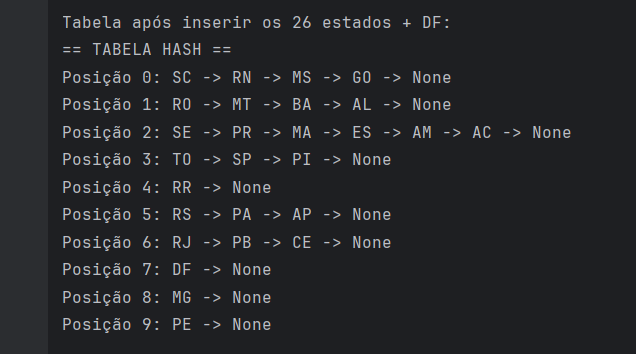
https://www.matematica.pt/util/resumos/tabela-ascii.php

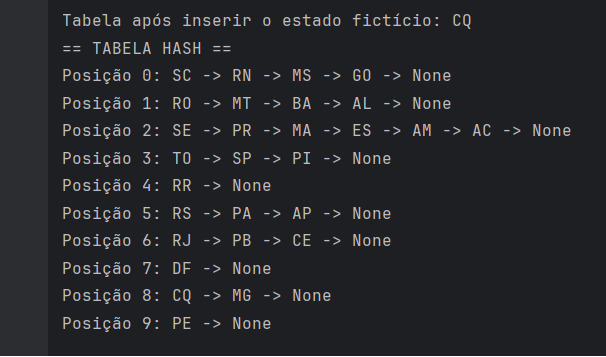
Apresentação de **Código da Questão 2**:

# Classe que representa cada estado como um nodo da lista  
class Estado:  
 def \_\_init\_\_(self, sigla, nome):  
 self.sigla = sigla  
 self.nome = nome  
 self.prox = None  
  
# Classe que representa a tabela hash com 10 posiçoes  
class TabelaHash:  
 def \_\_init\_\_(self):  
 self.tabela = [None] \* 10  
  
 # Função hash, com regra especial para DF  
 def funcaoHash(self, sigla):  
 if sigla == "DF":  
 return 7  
 else:  
 c1 = ord(sigla[0])  
 c2 = ord(sigla[1])  
 return (c1 + c2) % 10  
  
 # Inserir estado no início da lista encadeada  
 def inserir(self, sigla, nome):  
 pos = self.funcaoHash(sigla)  
 novo = Estado(sigla, nome)  
 novo.prox = self.tabela[pos]  
 self.tabela[pos] = novo  
  
 # Imprimir a tabela hash  
 def imprimir(self):  
 print("== TABELA HASH ==")  
 for i in range(10):  
 print(f"Posição {i}: ", end="")  
 atual = self.tabela[i]  
 while atual:  
 print(f"{atual.sigla}", end=" -> ")  
 atual = atual.prox  
 print("None")  
  
# ====== PROGRAMA PRINCIPAL COM TESTES ======  
if \_\_name\_\_ == "\_\_main\_\_":  
 tabela = TabelaHash()  
  
 # Imprimir tabela vazia  
 print("Tabela antes de inserir qualquer estado:")  
 tabela.imprimir()  
 print()  
  
 # Lista com os 27 estados (nome + sigla)  
 estados = [  
 ("AC", "Acre"), ("AL", "Alagoas"), ("AP", "Amapá"), ("AM", "Amazonas"),  
 ("BA", "Bahia"), ("CE", "Ceará"), ("DF", "Distrito Federal"), ("ES", "Espírito Santo"),  
 ("GO", "Goiás"), ("MA", "Maranhão"), ("MT", "Mato Grosso"), ("MS", "Mato Grosso do Sul"),  
 ("MG", "Minas Gerais"), ("PA", "Pará"), ("PB", "Paraíba"), ("PR", "Paraná"),  
 ("PE", "Pernambuco"), ("PI", "Piauí"), ("RJ", "Rio de Janeiro"), ("RN", "Rio Grande do Norte"),  
 ("RS", "Rio Grande do Sul"), ("RO", "Rondônia"), ("RR", "Roraima"), ("SC", "Santa Catarina"),  
 ("SP", "São Paulo"), ("SE", "Sergipe"), ("TO", "Tocantins")  
 ]  
  
 # Inserindo os 26 estados + DF  
 for sigla, nome in estados:  
 tabela.inserir(sigla, nome)  
  
 #Imprimi tabela após inserir 27 estados  
 print("Tabela após inserir os 26 estados + DF:")  
 tabela.imprimir()  
 print()  
  
 # Inserindo estado fictício com nome: Carlos Henrique Monnerat Quintanilha => CQ  
 tabela.inserir("CQ", "Carlos Henrique Monnerat Quintanilha")  
  
 # Imprimir tabela após inserir estado fictício  
 print("Tabela após inserir o estado fictício: CQ")  
 tabela.imprimir()

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

****

****

****