

ID: EA3-2025

Problema

Uma empresa de serviços em eletrônica pediu sua ajuda para gerenciar as demandas e funcionários. Desenvolva três TADs: `tTicket`, `tTecnico` e `tEletronica`, bem como uma `main` para executar as funcionalidades.

Os TADs devem ser **opacos**.

TAD `tTicket`: Um tipo `tTicket` possui um nome (`char *`), código de identificação (`int`), descrição (`char *`) e situação (`char *`). A situação do ticket pode ser `PENDENTE` ou `CONCLUIDO`.

O TAD `tTicket` possui as seguintes funções:

- **`tTicket* criaTicket(char* nome, int codigo, char *descricao)`:** aloca a memória de um tipo `tTicket` e o inicializa com os parâmetros passados, retornando o ponteiro alocado. Um ticket sempre é criado com a situação `PENDENTE`.
 - **`tTicket* leTicket()`:** lê as informações do ticket e retorna seu ponteiro alocado. Ordem de entrada: nome, código, descrição
 - **`alteraSituacaoTicket(tTicket *t)`:** altera a situação do ticket para `"CONCLUIDO"`
 - **`comparaTicket(tTicket *t, int codigo)`:** compara se o código passado por parâmetro corresponde ao código de identificação de um ticket.
 - **`void imprimeTicket(tTicket *t)`:** imprime as informações de um ticket a partir da seguinte formatação:
=====
- ```
Nome: Resistor queimado
Codigo: 1225
Situacao: PENDENTE
Descricao: Resistor de 190 Ohms de um circuito impresso está queimado
e precisa de substituicao
```
- **`destroiTicket(tTicket *t)`:** libera a memória de um tipo `tTicket`.

**TAD `Playlist`:** Um tipo `tTecnico` possui um nome (`char *`), `id` (`char *`), um vetor de ponteiros para tickets (`tTicket **`), uma quantidade de tickets atribuídos (`int`) e uma quantidade máxima de tickets a serem atribuídos (`int`)

O TAD `Tecnico` possui as seguintes funções:

- **`tTecnico *criaTecnico(char *nome, char *id)`:** cria um técnico a partir dos parâmetros. A lista de tickets inicia vazia e a quantidade máxima de tickets (`qtdMaxTickets`) começa com 1 e quantidade de tickets (`qtdTickets`) começa com 0.

- 
- **tTecnico \*leTecnico()**: cria um tecnico a partir das informações lidas. Ordem: nome. id.
  - **void atribuiTicket(tTecnico \*tec, tTicket \*tic)**: adiciona um ticket na lista de tickets de um tecnico.
  - **void aumentaCargaTecnico(tTecnico \*tec)**: dobra a qtdMaxTickets de um tecnico, fazendo uma realocação (realloc) no vetor tTickets correspondente a nova quantidade máxima de tickets
  - **void alteraSituacaoTicketTecnico(tTecnico \*tec, int codigo)**: altera a situacao de um ticket na lista de tickets de um tecnico para "CONCLUIDO", utilizando o codigo de identificacao para pesquisar o ticket.
  - **int quantidadeTicketsPendentes(tTecnico \*t)**: retorna a quantidade de tickets com a situacao "PENDENTE" da lista de tickets de um tecnico.
  - **void imprimeTecnico(tTecnico \*t)**: imprime as informacoes de um tecnico seguindo a formatação:  
Ficha do Tecnico  
Nome: Ronaldo  
Id: SUP0024  
Tickets atribuidos:  
=====
Nome: Resistor queimado  
Codigo: 1225  
Situacao: PENDENTE  
Descricao: Resistor de 190 Ohms de um circuito impresso está queimado e precisa de substituicao
  - **void destroiTecnico(tTecnico \*t)**: desaloca tTecnico.

**TAD tEletronica**: funções para manipulação dos tipo acima. Uma eletronica possui um nome (char \*), um vetor de ponteiros para tecnicos (tTecnico \*\*) e uma quantidade de tecnicos (int). Consulte o .h para definição dos TADs.

**main**: na main, comandos serão passados a fim de executar as implementações dos TADs. Você começará lendo e criando uma eletronica, e, após isso, receberá os comandos.

Dentre os comandos, se encontra:

- **CADASTRA**: lê e cria um tecnico, cadastrando-o na eletronica.
- **TICKET**: lê e cria um ticket, depois, lê um ID referente a um tecnico cadastrado numa eletronica. Por fim, atribui esse ticket ao tecnico.
- **AUMENTA**: lê um ID para um tecnico cadastrado numa eletronica. Aumenta o tamanho maximo de tickets suportado por esse tecnico.
- **CONCLUI**: lê um codigo para um ticket cadastrado numa eletronica e o conclui.

- 
- **IMPRIME**: imprime as informações de uma eletronica.
  - **FIM**: encerra o programa

Lembre-se de liberar toda a memoria alocada.



## Casos de Teste

Verifique os casos de entrada e saída para melhor compreender a formatação.