Tipo Abstrato de Dados (TAD) – Mala Direta

Uma empresa de mala direta utiliza um sistema informatizado para auxiliar no gerenciamento de informações sobre os destinatários de suas malas diretas. Essa empresa trabalha apenas com pessoas físicas. Nesse sistema, as informações referentes aos destinatários são armazenadas em um vetor de ponteiros para o TAD *Destinatario*.

O TAD *Destinatario* tem a seguinte interface (arquivo *Destinatario.h*):

```
typedef struct destinatario Destinatario;
```

Função	Objetivo
Destinatario *dest_cria_destinatario (char *nome, int d, int m, int a, char sexo, TEndereco ender, int instrucao, int profissao);	Aloca dinamicamente área para o tipo Destinatario, inicializa essa área com os valores recebidos como parâmetros e retorna o ponteiro para essa área (ou NULL se não foi possível alocá-la).
TData dest_consulta_nascimento (Destinatario *d);	Retorna a data de nascimento de determinada pessoa. O ponteiro para o destinatário é recebido como parâmetro. A data de nascimento é do tipo TData.
char *dest_consulta_nome (Destinatario *d);	Retorna o nome de determinada pessoa. O ponteiro para o destinatário é recebido como parâmetro. O nome do destinatário é representado como uma cadeia de caracteres.
Destinatario *dest_copia_destinatario (Destinatario *d);	Copia um destinatário, ou seja, aloca uma nova área dinamicamente, inicializa essa área com valores idênticos aos do destinatário recebido como parâmetro e retorna o ponteiro para a nova área (ou NULL se não foi possível alocá-la).
void dest_altera_endereco (Destinatario *d, TEndereco ender);	Altera o endereço de determinado destinatário. O ponteiro para o destinatário e seu novo endereço são recebidos como parâmetros.
void dest_altera_nivel_instrucao (Destinatario *d, int instrucao);	Altera o nível de instrução de determinado destinatário. O ponteiro para o destinatário e seu novo nível de instrução são recebidos como parâmetros.
int dest_verifica_aniversariante (Destinatario *d, int d, int m);	Compara uma data, fornecida como parâmetro, com a data de nascimento de um destinatário. O ponteiro para o destinatário e o dia e mês a serem comparados são recebidos como parâmetros. Retorna 1, se as datas são iguais, ou 0, em caso contrário.
int dest_verifica_profissao (Destinatario *d, int profissao);	Compara uma profissão, recebida como parâmetro, com a profissão de um destinatário. O ponteiro para o destinatário e a profissão a ser comparada são recebidos como parâmetros. Retorna 1, se as profissões são iguais, ou 0, em caso contrário.
void dest_libera_destinatario (Destinatario *d);	Libera a área de um destinatário cujo endereço é recebido como parâmetro.

onde *TData* corresponde ao tipo estruturado descrito abaixo.

```
struct tdata {
      int dia, mes, ano;
} ;
typedef struct tdata TData;
```

onde *TEndereco* corresponde ao tipo estruturado descrito abaixo.

```
struct tendereco {
      char logradouro [101];
      char numero[11];
      char complemento [51];
      char CEP[9];
} ;
typedef struct tendereco TEndereco;
```

Função Auxiliar

Questão 1 (1 ponto): Implemente em C a função *ComparaDatas* que:

- recebe duas variáveis (*d1* e *d2*) do tipo estruturado *TData*;
- retorna:
  - → -1, se a primeira data antecede cronologicamente a segunda data;
  - → 1, se a primeira data vem depois da segunda data;
  - → 0, se as duas datas são iguais.

O protótipo dessa função é o seguinte:

```
int ComparaDatas (TData d1, TData d2);
```

Usando o TAD

Questão 2 (5,0 pontos): Usando o TAD *Destinatario,* implemente em C a função *CadastraPessoa* com as seguintes características:

1. A função *CadastraPessoa* recebe como parâmetros*:*
   - um vetor de PONTEIROS para o TAD *Destinatario* (*vpd*) e o número de elementos desse vetor (*n*).
     - → O vetor de destinatários (*vpd*) está ordenado (ordem crescente) pela data de nascimento das pessoas.
     - → Ao vetor NÃO necessariamente está completamente preenchido. Suas posições finais podem estar preenchidas com NULL.
   - os dados de um novo destinatário:
     - → *nome*: nome da pessoa
     - → *dt_nasc*: sua data de nascimento
     - → *sexo*: seu sexo
     - → *ender*: seu endereço
     - → *instr*: seu nível de instrução
     - → *prof*: sua profissão

2. A função *CadastraPessoa* retorna*:*
   - → 1, se incluiu a nova pessoa no vetor com sucesso, ou 0, em caso de insucesso (não há espaço no vetor ou não se conseguiu alocar espaço para o novo destinatário).
     - A inclusão do novo destinatário no vetor deve preservar a ordenação, ou seja, o vetor tem que permanecer ordenado pelas datas de nascimento.
     - Use a função auxiliar implementada na questão 1 (*ComparaDatas*) para encontrar a posição onde o novo destinatário deve ser incluído.
   - → A quantidade de pessoas anteriormente cadastradas que façam aniversário na mesma data (dia e mês) do novo destinatário, através do ponteiro *aniversariantes*.

3. O protótipo da função *CadastraPessoa* é o seguinte:

```
int CadastraPessoa (Destinatario **vpd,
       int n, char *nome, TData dt_nasc, char sexo, TEndereco ender, int instr, int prof,
       int *aniversariantes);
```

Implementando o TAD

Questão 3 (1,0 pontos): Crie o arquivo de interface do TAD *Destinatario*. O nome do arquivo deve ser *Destinatario.h*.

Questão 4 (3,0 pontos): Implemente as seguintes funções do TAD *Destinatario* no arquivo *Destinatario.c*.

1. dest_cria_destinatario
2. dest_consulta_nascimento
3. dest_verifica_aniversariante