Trabalho Estrutura de Dados I – 2022/2
Universidade Federal do Pampa

1. Você deve implementar uma mini rede social! A rede social tem uma lista de usuários (representado pela estrutura users). Cada usuário tem um perfil (representado pela estrutura perfil) que determina as informações pessoais.

   Você deve implementar as estruturas e funções básicas para operar em listas, filas e pilhas caso haja necessidade. Utilize como base os códigos vistos em aula!

   O que entregar? Código fonte em C comentado. Códigos mal comentados, sem identação, copiados serão anulados.

   Como será avaliado? Cada item abaixo vale 1 ponto no máximo. A avaliação será dada pela corretude e clareza do código. Vocês podem sugerir alterações nos protótipos das funções e das estruturas desde que acompanhados de justificativa.

   Data da entrega? 26 de janeiro até às 18h30m
   O que apresentar? Apresentar brevemente no dia 26 o código e o funcionamento
   Quem entrega? Um membro da dupla.

```
struct filaRequisicoes{
      int id;                                    //identificação do
usuário;
      struct file *proximo;
};

struct perfil{
     int id;                                   //identificação do usuário;
     char* nome;
     ... //adicionar outras informações relevantes
};
typedef struct perfil Perfil;


//lista de usuários
struct user{
     struct perfil *perfilDoUsuario;            //informações do usuario
     struct perfil *listaDeAmigos;         //lista de amigos já confirmados
     struct filaRequisicoes *amigosPendentes;  //fila de amigos ainda não confirmados
     struct user *proximoUser;        //ponteiro para o próximo usuario da lista
};
typedef struct users User;
```

A sua rede social deve implementar funções básicas para:

a) Adicionar/criar um novo usuário na rede social. A função deve criar o usuário e adicionar a lista de usuários ativos. Por padrão, o usuário criado não tem amigos. O **id** do usuário deve ser um número sequencial único por usuário.

```
void criarUsuario(char *nome, ... , );
```

b) Buscar um perfil na rede social pelo nome ou pelo id. A função deve retornar o ponteiro para o perfil. Caso não encontre, retornar Null.

```
User* buscaPorId(User *listaUser, int id);
User* buscaPorNome(User *listaUser, char *nome);
```

c) Remover um usuário da rede social pelo nome ou pelo id. A função deve retornar 1 caso a remoção seja realizada, e 0 caso contrário.

```
int removerPorId(User *listaUser, int id);
int  removerPorId(User *listaUser, int id);
```

d) Solicitar a amizade de um perfil. A solicitação de amizade deve ser adicionada a fila de solicitações do perfil referenciado pelo idPerfilAmigo.

```
void solicitarAmizade (User *listaUser, int idPerfilSolicitante, int idPerfilAmigo);
```

e) Aceitar ou rejeitar solicitação de amizade. A implementação deve respeitar a ordem da fila de solicitações. Quando uma solicitação é aceita, o perfil deve ser adicionado a lista de amigos permanentes.

```
void aceitarPrimeiraSolicitacaoAmizade (User *listaUser, int idPerfil);
void aceitarTodasSolicitacaoAmizade (User *listaUser, int idPerfil);
void rejeitarTodasSolicitacaoAmizade (User *listaUser, int idPerfil);
void rejeitarPrimeiraSolicitacaoAmizade (User *listaUser, int idPerfil);
```

f) Função para retornar o número de amigos de um perfil e da rede social.

```
int numAmigos(User *listaUser, int idPerfil);
int numUsers(User *listaUser);
```

g) Função para retornar o número de solicitações de amizades de um perfil.

```
int numSolicitacoesAmigos(User *listaUser, int idPerfil);
```

h) Função para determinar o perfil que é mais amigo de todos. Ou seja, aquele perfil que esteja mais recorrente na lista de usuários.

```
User*  quemEhOPerfilMaisAmigo(User *listaUser);
```

i) Função para realizar o “match” entre perfis. A ideia do “matching” é sugerir ao usuário perfis ainda não amigos para serem adicionados. A função deve retornar um perfil. Como métrica de similaridade, você pode utilizar a quantidade de amigos em comum.

```
User* recomendarAmizade(int idPerfil, User *listaUser);
```

j) Implementar corretamente o uso de listas, filas e pilhas. Atentar para o uso correto das estruturas e da alocação e desalocação dinâmica de memória.