Universidade Federal do Espírito Santo – Departamento de Informática
Estruturas de Dados I (INF09292)

1º Trabalho Prático

Período: 2022/2

Profa Patrícia Dockhorn Costa
E-mail: pdcosta@inf.ufes.br

Data de Entrega: 21/11/2022 - Trabalho em Dupla

Este trabalho tem como objetivo praticar o uso de tipos abstratos de dados e estruturas do tipo Lista.

Regras Importantes

● Não é tolerado plágio. Trabalhos copiados serão penalizados com zero. ● A data de entrega é inadiável. Para cada dia de atraso, é retirado um ponto da nota do trabalho.

Material a entregar

- PDF : Documentação do trabalho, que deve conter:
  - Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  - Implementação: descrição da implementação do programa. Devem ser detalhadas as estruturas de dados utilizadas (dê preferência a diagramas ilustrativos), o funcionamento das principais funções utilizadas, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
  - Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  - Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet, se for o caso.

- Pelo Classroom (**arquivos zipados!**):
  - Todos os arquivos .c e .h criados (exigido código muito bem documentado!).
  - O makefile.
  - Lembrando de modularizar o seu programa usando Tipos Abstratos de Dados (TAD's), como discutido nos vídeos e nas aulas síncronas (com

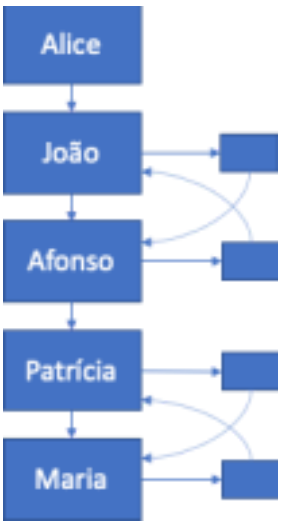
tipos opacos).

EdMatch!

Atualmente, as redes sociais ocupam um lugar de destaque nas vidas das pessoas. É por meio delas que as pessoas interagem, consomem e produzem informações, mantêm laços afetivos, expressam emoções e formam opinião. Reconhecendo a importância deste domínio, o primeiro trabalho de ED propõe a implementação, simplificada, de uma rede social chamada EdMatch.

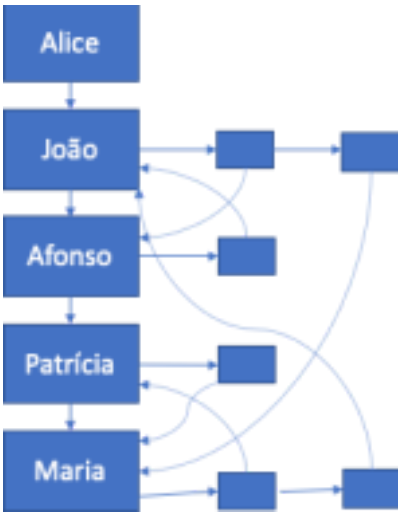
As amizades entre usuários da rede EdMatch devem ser formadas automaticamente, quando duas pessoas indicam que gostam uma da outra, ou seja, quando "dão um like". Considere o seguinte cenário de utilização desta rede social:

Alice, João, Patrícia, Afonso e Maria são usuários da rede EdMatch. Logo de início, João e Afonso, como são amigos na vida real, já indicaram o "like" e, por isso, são amigos na EdMatch. A amizade é recíproca e, portanto, se João é amigo de Afonso, Afonso também é amigo de João. O mesmo acontece com Patrícia e Maria. Já Alice, neste primeiro momento, ainda não tem amigos na EdMatch. No mundo das estruturas de dados, essa rede de amizades pode ser implementada como uma lista encadeada de listas encadeadas[1], como mostrado na Figura 1. Nessa Figura, a lista principal representa os usuários cadastrados na EdMatch. Já as listas "secundárias" representam as amizades.



Considere agora que, em um outro momento, João tenha dado um "like" em Maria e ela, eventualmente, também deu um "like" em João, formando, assim, a amizade entre eles. A estrutura de dados que representa essa visão atualizada da EdMatch pode ser vista na Figura 2. Eventualmente, os usuários também podem dar um "unlike" em algum amigo, desfazendo, portanto, uma amizade já estabelecida.

[1] Tecnicamente chamada de "lista de adjacência"



Na EdMatch, os usuários podem fazer postagens de textos que serão compartilhadas apenas com seus amigos.

As interações dos usuários com a rede EdMatch serão simuladas por meio de arquivos de entradas. Enquanto que, na realidade, as interações entre usuários e redes sociais são assíncronas, aqui no EdMatch as interações serão de forma síncrona, em ciclos formados pela leitura de linhas dos arquivos de entrada, como explicado na seção a seguir.

Entradas do Sistema

A primeira entrada a ser considerada consiste no **número de ciclos** a ser considerado na execução do EdMatch e deve ser fornecido por meio de **argc e argv**[2], por exemplo:

./EdMatch.out 10, onde 10 é o número de ciclos que devem ser considerados nessa execução do EdMatch.

O primeiro arquivo de entrada a ser lido pelo EdMatch deve ser o arquivo usuarios.txt, no qual são especificados todos os usuários cadastrados na rede social. Para simplificar, podemos considerar apenas nomes simples, sem acentuação ou caracteres especiais. Cada nome corresponde a um usuário, seguido por mais algumas informações iniciais de cada um. As informações iniciais são idade, onde mora e alguns de seus hobbies. A formatação será da seguinte forma:

```
nome;idade;localização;<lista de hobbies separados por vírgula>
```

Um usuário pode ter um ou mais hobbies, caso tenha mais de um, será separado por vírgulas.

[2] Vimos argc e argv em sala de aula. Caso tenham perdido a explicação, sugiro olhar os vídeos da primeira aula e/ou procurar material na internet.

Arquivo *usuarios.txt*:

```
Alice;25;Vitoria-ES;artes,guitarra,carro
Joao;23;Vitoria-ES;bike,carro
Afonso;19;Viana-ES;computador,violao
Patricia;42;Vitoria-ES;bike,computador
Maria;34;DomingosMartins-ES;academia
```

Cada usuário cadastrado deve ter um arquivo de entrada especificando suas interações com o sistema. As interações são representadas por ciclos de execução, uma por linha do arquivo, e são chamadas de *pacotes de dados*. Por exemplo, se foram definidos 10 ciclos de execução, todos os arquivos de usuários precisam ter, necessariamente, 10 linhas, ou *10 pacotes*. A padronização dos pacotes é feita da seguinte forma:

```
<like>;<unlike>;<mudança do campo hobbies>;<mensagem de post>
```

Explicando cada item do pacote:

- O campo "like" especifica o nome do usuário para o qual este usuário quer indicar um "like";
- O campo "unlike" especifica o nome do usuário para o qual este usuário quer indicar um "unlike";
- Caso o campo “mudança de hobbies” esteja preenchido, será feita uma "repaginada" nos hobbies do usuário, excluindo todos os hobbies antigos e colocando os novos. A listagem tem a mesma formatação de hobbies, que é apresentada no *usuarios.txt (separados por vírgulas simples) .*
- A "mensagem" especifica o texto a ser enviado para os amigos desse usuário (considerando as amizades existentes antes deste ciclo). Caso não haja nada escrito neste campo, é considerado que o usuário não está postando nenhuma mensagem naquele ciclo.

O arquivo, a seguir, descreve as entradas dos usuários Afonso, Joao e Maria, caso haja 5 ciclos nesta execução particular do EdMatch; quando o campo for vazio, será representado por um ponto final:

Afonso.txt

```
Joao;.;.;.
.;.;.;.
.;.;.;ola
Maria;.;.;.
.;.;.;hoje tem festa
```

Joao.txt

```
Afonso;.;.;.
 .;.;.;ola amigos!
 .;.;.;estou feliz!
Maria;.;.;que chato…
```

```
 .;.;bike,carro,novela;final da novela hoje!
```

Maria.txt

```
 .;.;.;.
Joao;.;.;mensagem de Maria
 .;.;.;outra mensagem
 .;.;.;.
 .;.;.;sou muito legal
```

É importante notar que os "likes" devem ficar "armazenados" nas suas estruturas de dados. Considere o seguinte cenário, no qual cada número representa um ciclo de execução:

1. Fulano deu like em Siclano e Siclano deu like em Fulano -> amizade formada; 2. Fulano deu dislike em Siclano -> amizade desfeita (porém, o like de Siclano para Fulano deve ficar armazenado);
3. Fulano deu like em Siclano -> amizade formada;
4. Fulano deu deslike em Siclano e Siclano deu deslike em Fulano -> amizade desfeita;
5. Fulano deu like Siclano -> nada acontece.

Saídas do Sistema

O programa gera dois arquivos de saída com diferentes propósitos: um arquivo de relatório (report), e um arquivo de logs. O arquivo de Logs descreve cada ação que está acontecendo na rede social como um todo de forma síncrona, utilizando diferentes símbolos para demonstrar as suas operações. A tabela a seguir mostra a padronização que deve ser adotada para cada ação possível:

+	Curtida (like)	**#**	Amizade criada
_	Descurtida (unlike)	**$**	Amizade desfeita
*****	Publicou	**!**	Mudança de hobbie
->	Conteúdo da publicação	**~**	Conteúdo da mudança de hobbie

A seguir está o arquivo de logs para o exemplo anterior.

logs.txt

```
+ Joao curtiu Kevin
_ Joao descurtiu Kevin
+ Afonso curtiu Caio
* Afonso publicou:
-> e ae galera, tudo bem?
+ Caio curtiu Afonso
# Caio e Afonso viraram amigos
+ Kevin curtiu Joao
* Kevin publicou:
-> Brabeza
+ Afonso curtiu Joao
! Afonso mudou seu hobbie para:
~ mountain bike
* Afonso publicou:
-> cheguei cansado hoje
* Caio publicou:
-> Bom dia rapeize!!!
! Kevin mudou seu hobbie para:
~ programar
```

Caso Afonso tivesse descurtido Caio em algum período no qual eles eram amigos, a mensagem que iria ser mostrada nos logs seria:

```
_ Afonso descurtiu Caio
$ Afonso desfez amizade com Caio
```

O arquivo reports, por sua vez, apresenta uma forma de sintetizar toda a informação que foi processada no programa de uma maneira mais apresentável, listando cada usuário em sua ordem de chamada no início do programa, com suas informações mais importantes. As informações básicas (Nome, idade, localização) e informações que são mais complexas, como hobbies (que podem mudar pelo programa), posts publicados (que devem mostrar o alcance de cada post individualmente), posts recebidos (com seu autor), matches (que são feitos e desfeitos pelo programa) e por último uma sugestão de amizade (a ser explicado mais para frente no documento).

Mostraremos um exemplo de um arquivo report, para facilitar a compreensão do código:
reports.txt

```
====================
Joao
23 anos
Vitoria-ES
```

0 amigos

HOBBIES:
bike
carro

POSTS PUBLICADOS

POSTS DO FEED

AMIGOS

SUGESTOES DE AMIZADE
Afonso
Caio

==========
Afonso
19 anos
Vitoria-ES
1 amigos

HOBBIES
mountain bike

POSTS PUBLICADOS
"e ae galera, tudo bem?" alcance:

0 "cheguei cansado hoje" alcance:

1

POSTS DO FEED
"Bom dia rapeize!!!" - Caio

MATCHES
Caio

SUGESTOES DE AMIZADE
Joao

==========
Caio

```
20 anos
Vitoria-ES
1 amigos

HOBBIES
carro

POSTS PUBLICADOS
"Bom dia rapeize!!!" alcance: 1

POSTS DO FEED
"cheguei cansado hoje" - Afonso

MATCHES
Afonso

SUGESTOES DE AMIZADE
Joao

====================
Kevin
21 anos
DomingosMartins-ES
0 amigos

HOBBIES
programar

POSTS PUBLICADOS
"Brabeza" alcance: 0

POSTS DO FEED

MATCHES

SUGESTOES DE AMIZADE
```

Sugestões de Amizade

O sistema de sugestões de amizade consiste em uma tentativa do programa de criar novas amizades entre usuários que não são amigos no final do programa. Não necessariamente eles não eram amigos antes, visto que esse armazenamento é muito

custoso para o sistema. Para o sistema recomendar uma amizade, os usuários precisam satisfazer TODOS os seguintes atributos:

- Possuir a mesma localização;
- Uma diferença de idade de no máximo 5 anos;
- Não serem amigos.

No exemplo mostrado acima, podemos ver que na saída de João foi recomendado a usuária Alice, O motivo não é especificado, mas para uma maior compreensão analisaremos a recomendação neste caso, de acordo com a tabela a seguir:

	Idade	Localização	Hobbies
João	23	Vitoria-ES	bike,carro
Alice	25	Vitoria-ES	artes,guitarra,carro

A diferença de idade entre João e Alice é 2, eles têm a mesma localização, e não são amigos. Logo, a recomendação é válida.

Posts

Posts são uma maneira de comunicação entre pessoas que EdMatch deu *match*. Quando o usuário manda uma mensagem, ela é repassada para todas as pessoas que são atualmente amigos com ela, seguindo a ordem de processamento explicada a seguir. O post fica salvo na área de posts recebidos de cada usuário e os posts que o próprio usuário enviou, devem ficar salvos no campo *posts enviados*. No posts recebidos é mostrado o conteúdo do post e quem o enviou; já no post enviados, é mostrado o conteúdo e o “alcance” do post, que consiste na quantidade de usuários que recebeu tal post (o autor não é contabilizado nessa contagem).

O sistema de gerenciamento de Posts é feito por ordem de processamento, na qual a ordem em si é ditada pela ordem de listagem de usuários no arquivo usuarios.txt, onde todas as informações do usuário são processadas antes de passar para o próximo. Para demonstrar tal processo, mostraremos um exemplo extra.

Arquivo *usuarios.txt*:

```
Afonso;19;Viana-ES;computador,violao
Joao;23;Vitoria-ES;bike,carro
```

Como Afonso foi listado primeiro e João em segundo, esta deve ser a ordem de execução dos usuários. Dentro de cada usuário, as informações são processadas da esquerda para direita (likes;unlikes;hobbies;post).

Logo, se um usuário der *unlike* antes do post, a pessoa que ele deu unlike terá a amizade desfeita antes de postar, não recebendo o post. De maneira similar, se o usuário que já tinha um like, e este retribui o *like* e, posta na mesma linha (mesmo pacote), a amizade será formada e o post deve ser recebido.

Porém, se dois usuários se deram like na mesma linha, e o primeiro a dar like postar algo naquela linha, o post não chegará para o usuário a ser processado depois, pois o post do primeiro usuário é processado antes do like do segundo usuário.

No exemplo a seguir, todos os posts que não chegam para o outro usuário estão marcados em vermelho, já os que chegam são marcados em verde.

Afonso.txt

```
Joao;.;.;Mensagem 01
.;.;.;Mensagem 02
.;Joao;.;Mensagem 03
Joao;.;.;Mensagem 04
.;.;.;Mensagem 05
```

Joao.txt

```
Afonso;.;.;Post 01
.;Afonso;.;Post 02
Afonso;.;.;Post 03
.;Afonso;.;Post 04
Afonso;.;.;Post 05
```

Regras Importantes

- Usar listas encadeadas para implementar a EdMatch;
- Estruture/modularize EdMatch usando as técnicas de TAD ensinados em sala de aula;
- Mantenha as saídas seguindo o padrão descrito aqui nessa especificação (testes serão automatizados na medida do possível);

BOM TRABALHO!!!