Universidade Federal do Pampa
Estrutura de Dados II
Prof. Marcelo Caggiani Luizelli
Trabalho Complementar
Entrega: 07 de dezembro de 2018

Regras para a execuçãoo do trabalho:

1. O peso do trabalho corresponde a 1,5 pontos na média final.
2. O trabalho será realizado individualmente.
3. O aluno, até o prazo final de entrega, deverá enviar **via Moodle** os arquivos de código-fonte do programa (arquivo com terminação `.c/.cpp/.java`).
4. O arquivo deve possuir um cabeçalho contendo o nome completo do aluno e respectivo número de matrícula, além de uma breve descrição do programa. O código deve estar corretamente identado. O código deve estar devidamente comentado para facilidade de entendimento (comentários sobre os tipos de dados utilizados, o que faz cada função, qual o tipo de cálculo que está sendo realizado, etc).
5. Política de plágio: nota integralmente zerada.

Sobre a avaliação do trabalho:

1. No dia **07/12/2018** às **23:59** encerra o prazo de entrega do trabalho complementar.
2. O programa deve ao menos **compilar** para ser avaliado. Programas que não compilam não serão avaliados.

Datas:

1. **Apresentação do enunciado do trabalho**: 09/12/2018
2. **Entrega do trabalho**: 07/12/2018 até as 23:59 (via Moodle).

Enunciado:

Desenvolva uma versão reduzida de uma rede social. Você deve projetar e implementar a estrutura de dados para permitir um conjunto de operações na rede social. A rede social deve ser composta por um conjunto de perfis e deve permitir o inter-relacionamento entre eles. Utilize a estrutura de dados mais adequada para a representação.

Operações que devem ser desenvolvidas:

1. Criação de um perfil: o perfil deve conter um `id` (int) e um `nome` (string). O protótipo da função deve ser `int addPerfil(int id, char* nome)`. O retorno indica se o perfil foi ou não adicionado com sucesso. Perfis duplicados não são permitidos.

2. Adição de amigos ao perfil: a estrutura de dados deve permitir que um perfil A estabeleça um relacionamento com um perfil B. O protótipo da função deve ser `int addAmigo(int idPerfil1, idPerfil2)`.

3. Remoção de amigos do perfil: a estrutura de dados deve permitir que um perfil A desestabeleça um relacionamento com um perfil B. O protótipo da função deve ser `int removeAmigo(int idPerfil1, idPerfil2)`.

4. Buscar um perfil na rede social: a estrutura de dados deve permitir a busca global por um determinado perfil na estrutura de dados. O protótipo da função deve ser `Perfil buscaPerfil(char *nome)`.

5. Busca um perfil dentre os amigos: a estrutura de dados deve permitir a busca de um determinado perfil entre os relacionamentos já estabelecidos. O protótipo da função deve ser `Perfil buscaPerfilAmigos(int idPerfil1, char *nome)`.

6. Buscar um perfil dentre os amigos dos amigos: a estrutura de dados deve permitir a busca de um determinado perfil dentre os amigos dos amigos. O protótipo da função deve ser `Perfil buscaPerfilAmigosAdj(int idPerfil1, char *nome)`.

7. Buscar um perfil em níveis: a estrutura de dados deve permitir a busca de um determinado perfil dentre os amigos dos amigos. Observe que o nível representa o nível máximo de uma busca em largura. O protótipo da função deve ser `Perfil buscaPerfilAmigosNivel(int idPerfil1, char *nome, int nivelMax)`.

8. Verificar amigos em comum: a estrutura deve permitir comparar amigos em comum entre perfis e retornar a interseção. O protótipo da função deve ser `Lista<Perfil> comparaAmigos(int idPerfil1, int idPerfil2)`.

9. Sugestão de amigos: desenvolva um algoritmo para sugerir amigos. O protótipo da função deve ser `Lista<Perfil> sugerirAmigos(int idPerfil1)`.

10. Calcular distância entre perfis: a estrutura deve permitir computar a distância (em saltos) entre os perfis que ainda não estão conectados. Se um perfil A está conectado diretamente a um perfil B, então a distância entre eles é zero. O protótipo da função deve ser `int distanciaEntrePerfis(int idPerfil1, int idPerfil2)`.

11. Verificar grupos de amizade: a estrutura deve permitir computar quais são os grupos de amigos na rede social. Um grupo de amigos é aquele definido por um componente fortemente conexo. O protótipo da função deve ser similar a `List<Grupos> gruposAmizade()`.

12. Fusão de perfis: a estrutura deve permitir fundir dois ou mais perfis em um novo perfil. O protótipo da função deve ser similar a `Perfil mergePerfis(List<Perfil> listaPerfil)`.

13. Detectar perfil falso: a estrutura deve permitir identificar perfis potencialmente falsos. Um perfil pode ser considerado falso se o nível de relacionamento dele com os demais perfis da rede social é considerado baixo (ou abaixo de um determinado limiar – e.g., a média). O nível de relacionamento pode ser medido pelo número de amigos (relacionamentos) estabelecidos. O protótipo da função deve ser `Lista<Perfil> buscaPerfilFalso()`.

14. Detectar perfis influenciadores: a estrutura deve permitir identificar quais perfils são potencialmente influenciadores na rede social. Um determinado perfil pode ser considerado influenciador se o nível de relacionamento dele com os demais perfis da rede social é considerado alto em comparação com um limiar.

A estrutura projetada deve ser validada com o conjunto de dados disponíveis em no Moodle para datasets do Facebook e Orkut. Em ambos os arquivos, existe uma lista de identificação de perfis conectados.