ESCOLA DE ENGENHARIA
Universidade Federal de Minas Gerais

Departamento de Engenharia Elétrica
Programação Orientada a Objetos – 1º Semestre de 2014
Profª. Raquel Mini

1º TRABALHO PRÁTICO

VALOR: 20 PONTOS

Observações:

- ✓ O trabalho pode ser executado **em grupo de no máximo 2 alunos.** Também pode ser executado individualmente.
- ✓ É permitido discutir os problemas e estratégias de solução com seus colegas.
- ✓ Quando se tratar de escrever ou implementar as soluções, isto deve ser feito **apenas com o componente do seu grupo.**
- ✓ Se utilizar qualquer fonte externa para implementar suas respostas, você deve **citar** estas fontes: artigos ou livros, amigos ou colegas, informação que você encontrou na Internet – **qualquer coisa em qualquer lugar**!
- ✓ É melhor tentar solucionar os problemas você mesmo, pois **solucionar problemas é um componente fundamental** neste curso. Não vamos penaliza-lo se você utilizar uma ajuda externa, desde que devidamente citada, e desde que esta ajuda não seja a cópia do trabalho de um colega.
- ✓ **Utilizar o trabalho dos outros, como se fosse seu, é plágio ⇒ desonestidade acadêmica será punida com severidade.**
- ✓ A entrega dos trabalhos será feita no Moodle.
- ✓ A data limite de entrega é o dia **15/04/2014**.
- ✓ Todas as questões devem ser muito bem documentadas. Ao criar uma classe, crie a documentação correspondente, o mesmo valendo para atributos, métodos, casos de uso, etc...
- ✓ Organize corretamente seu trabalho: crie os arquivos de interface (.h) e de implementação (.cpp) para cada classe e um arquivo separado para o programa principal (main.cpp).

Questões:

1. (6 pontos) Uma lista simplesmente encadeada é uma estrutura de dados muito utilizada quando se trabalha com dados que podem crescer dinamicamente. Cada nó de uma lista encadeada possui um campo para armazenar dados e um ponteiro para o próximo nó. O último nó da lista aponta para NULL. Para acessar o início da lista, existe um ponteiro para a sua cabeça (head). Crie uma classe para uma lista encadeada de inteiros que implemente a seguinte interface básica:

(a) Crie uma lista vazia (Construtor);
(b) Insere um novo inteiro na lista. Os inteiros devem ser armazenados ordenados na lista, ou seja, cada novo inteiro deve ser inserido de tal modo que fique entre o inteiro inferior a ele e o superior a ele já presentes na lista. Se houver a tentativa de inserção de um inteiro já presente na lista, ele não deve ser inserido;
(c) Dado um inteiro, identifique se ele está presente na lista;
(d) Dado um inteiro, retire-o da lista, se ele estiver presente nela;
(e) Grave os elementos em uma ostream (que pode ser cout ou uma ofstream) recebida como parâmetro);
(f) Leia os elementos da lista de uma ifstream (que pode ser cin ou uma ifstream) recebida como parâmetro.
(g) Crie um Destrutor que desaloque todos os nós ainda existentes na lista no momento de sua destruição.

Escreva um programa de teste da lista.
Dicas: (a) consulte o livro de AEDs II para eventuais dúvidas. (b) Você pode criar uma classe aninhada para encapsular o conceito de nó da lista.

2. (7 pontos) Crie uma classe JogoDaVelha que permita escrever um programa completo sobre o jogo do mesmo nome. A classe deve armazenar todas as possíveis posições do tabuleiro. Ela também gerenciará o jogo, permitindo que os jogadores efetuem suas jogadas (para cada jogada é necessário passar para o jogo qual o jogador, se o primeiro ou o segundo, e qual a posição em que ele está jogando). A função de jogada deve testar se ela é possível (a posição ainda não está ocupada). A classe deve ter um método para testar, após cada jogada, se o jogador que acabou de jogar venceu. Também deve indicar, caso o jogo termine empatado, esta situação. Crie um método para imprimir na tela a situação atual do jogo (use caracteres alfanuméricos para isso). Crie um método que retorne a situação atual do jogo (quais casas estão ocupadas e por qual jogador). Crie um programa que use esta classe e que permita que um usuário jogue contra o computador. Crie uma classe, "Jogador Virtual", que será o adversário de um jogador humano. O jogador virtual receberá do jogo da velha a situação atual do jogo e escolherá sua próxima jogada. Você pode fazer com que as escolhas do jogador virtual sejam aleatórias, ou então usar alguma técnica para fazer com que as jogadas desse jogador sejam bem escolhidas.

3. (7 pontos) Crie uma classe denominada IntegerSet. Cada objeto da classe pode armazenar inteiros na faixa de 0 a 100. O conjunto é representado internamente por um array de ***bool***. O elemento a[i] é true se o inteiro i estiver no conjunto. O elemento a[j] é false se o inteiro j não estiver no conjunto. O construtor default inicializa o conjunto

como um conjunto vazio, isto é um array com todas as 101 posições iguais a false. Forneça funções membro para as operações comuns com conjuntos. (i) forneça a função ***uniao***, que cria um terceiro conjunto que é a união de dois conjuntos existentes (i.e., um elemento do array do terceiro conjunto é true se estiver presente no primeiro *ou* no segundo conjunto. (ii) forneça a função membro intersecao, para efetuar a interseção entre dois conjuntos; (iii) forneça a função membro insereElemento(int k), que insere o inteiro k e a função deleteElemento(int k) que remove o inteiro k do conjunto. (iv) Forneça a função imprime que imprime os elementos presentes no conjunto na saída padrão separados por espaços. Imprima --- se o conjunto for vazio. (v) Escreva a função membro leDeArquivo(std::string nomeArquivo) que recebe o nome de um arquivo onde estão armazenados os elementos de um conjunto, lê os elementos e os insere no conjunto (vi) Forneça a função membro igual, que determina se dois conjuntos são iguais. (vii) Escreva um programa principal que teste se suas funções estão funcionando corretamente. O programa, no mínimo, deve ler dois conjuntos a partir de dois arquivos e efetuar as operações entre eles.