NORMAS PARA ENTREGA E AVALIAÇÃO DO TRABALHO 2

1. O trabalho é individual e consiste na implementação de um **Vetor Encadeado** com tipo abstrato de dados.
2. O Vetor Encadeado começa inicialmente vazio e terá seu tamanho incrementado ou decrementado na exata quantidade de elementos que ele possui.
3. O trabalho segue o seguinte padrão:
    a. De *main.cpp:*
        i. A estrutura básica do código *main.cpp* está disponível no SIGAA.
        ii. Você pode criar outros arquivos *.cpp e/ou .h* e incluí-los ao *main.cpp*;
        iii. Deve incluir a classe Vetor.
        iv. A função *main()* é caso de teste e referência para o uso de Vetor;
        v. *A função main()* deve executar corretamente se não alterado;

    b. Da Classe Vetor:
        i. Implementa o Vetor Encadeado;
        ii. Deve ser criada em arquivo próprio e incluído no *main.cpp*.
        iii. Pode conter quaisquer outros métodos *privados ou protegidos*, desde que os *públicos* sejam **somente**:
            1. **Vetor()** : construtor sem parâmetros que cria um Vetor Encadeado vazio;
            2. **~Vetor()**: destrutor que deleta toda memória alocada dinamicamente;
            3. **int size()**: retorna o número de elementos de Vetor
            4. **bool add(T i_)**: adiciona no fim do Vetor o elemento i_ do tipo T. Retorna TRUE caso o elemento seja adicionado corretamente, e FALSE caso contrário. Utilize TRY/CATCH;
            5. **bool remove(int i)**: remove o "i"-ésimo elemento de Vetor, começando pela i=0 (primeira posição do vetor). Retorna FALSE caso "i" seja uma posição inválida, ou caso a remoção gere erro; retorna TRUE caso contrário. Utilize TRY/CATCH;
            6. **void show()**: existe a lista na tela. Implementação livre;
            7. **T at(int i)**: retorna o elemento da i-ésima posição do vetor, começando com i=0 (primeira posição do vetor). Gera exceção caso "i" seja uma posição inválida;
            8. **bool sort (*func_compara*)**: ordena Vetor. Retorna TRUE caso o algoritmo seja ordenado, e FALSE caso o Vetor seja vazio. Uma vez que Vetor é um Template, ele não tem tipo definido. Assim, a "*func_compara"* é uma

função passada por referência como parâmetro do método sort. Ela é utilizada comparar elementos de sort.

iv. Os alunos podem adicionar novos atributos ou métodos em Vetor, desde que privados ou protegidos.

c. Das Classes e funções adicionais
   i. Deve ser criada em arquivo próprio e utilizada no código;
   ii. Se classe, a visibilidade dos atributos e métodos fica a critério do aluno;

4. Das Obrigações
   a. O aluno deve:
      i. Criar makefile completo que utiliza c++17 ou superior (necessário para executar templates);
      ii. Criar Arquivos .h/.cpp necessários para implementação correta do código;
      iii. Liberar todas as variáveis alocadas dinamicamente*;*
      iv. Criar código que execute corretamente o *main()* tal como fornecido no código base exemplo.

   b. O aluno não deve:
      i. Plagiar ou copiar código alheio;
      ii. Utilizar bibliotecas que resolvem o problema;

5. A entrega do trabalho final é considerada completa com:
   a. Entrega da implementação contendo os arquivos
      i. makefile;
      ii. main.cpp;
      iii. demais .h e .cpp criados;
      iv. Vídeo de no máximo 3 minutos explicando sucintamente o código e mostrando exemplos de execução.
         1. O vídeo pode ser enviado em anexo ou através de um link escrito em arquivo de texto enviado junto com os demais arquivos.
   b. Código compilando completamente através do comando *make;*

6. A nota será composto pela execução de diversos testes no main.cpp, tal como no código exemplo disponibilizado no SIGAA.
   a. Será considera nota 0,0:
      i. O não cumprimento de qualquer item ou subitem de 3, 4 e/ou 5;
      ii. Submissão dos arquivos por outro meio diferente do campo específico no SIGAA;
      iii. Submissão dos arquivos fora do prazo estipulado;
      iv. Erros de execução durante os testes;

7. Exceções e casos omissos devem ser tratados diretamente com o Professor da disciplina.