Nome: _____ Matrícula: _____

Leia atentamente as instruções abaixo:

- Fazer o download do arquivo `Avaliacao1EDLab2-2022-3.zip` do *site* do curso e descompactá-lo na sua máquina. Este arquivo contém todos os códigos para o desenvolvimento da prova.
- A resposta de cada questão deve, **obrigatoriamente**, estar entre cada par de marcadores (//Qi, //-Qi). Assim, a questão 1 está entre //Q1 e //-Q1, a questão 2 entre //Q2 e //-Q2 e assim por diante. Não remover, em hipótese alguma, tais marcadores de questões da sua prova. Caso sua solução tenha mais de uma função ou operação, elas devem estar entre esses marcadores.
- Colocar no arquivo `main.cpp` seu nome completo e número de matrícula.
- A prova é **individual** e **sem qualquer tipo de consulta**.
- Existe apenas um projeto do Code::Blocks que será usado na prova.
- Ao terminar a avaliação, enviar ao servidor – usando a janela de upload – cada arquivo de código que contém as respostas das questões da sua prova. Aguarde um momento e verá as suas respostas de cada questão da prova.
- **O desenvolvimento e envio do código são de inteira responsabilidade do aluno!**
- Endereço do servidor: `http://172.18.40.119:8080/edlab2ufjf/`

Questões:

1. (20 Pontos) Implemente a função `int* vetorNegativos(int vet[], int n, int *t)`, que recebe como parâmetros um vetor de números inteiros `vet` e seu tamanho `n`, e retorna um novo vetor com os elementos negativos de `vet`. O novo vetor deve ser alocado com o número adequado de posições. Se não houver nenhum valor negativo no vetor, a função deve retornar `NULL`. O tamanho do novo vetor deve ser armazenado em `t`.

```
int* vetorNegativos(int vet[], int n, int *t);
```

2. (20 Pontos) Implemente a função `bool verificaIguais(int vet1[], int vet2[], int n)`, que recebe como parâmetros dois vetores de inteiros `vet1` e `vet2` de mesmo tamanho `n`. A função deve verificar **recursivamente** se os dois vetores são iguais (possuem os mesmos valores). Caso não sejam iguais, a função deve retornar `false`. Caso os vetores sejam iguais, a função deve retornar `true`.

```
bool verificaIguais(int vet1[], int vet2[], int n);
```

3. (30 Pontos) A seguir encontra-se a classe que implementa o TAD `Avaliacao`. Os dados armazenados para representar uma avaliação são:

   - `int questoes` (número inteiro positivo indicando o total de questões de uma atividade);
   - `float valor` (valor das questões);
   - `float *notas` (vetor contendo as notas obtidas em cada questão).

   Para o TAD `Avaliacao`, desenvolver:

   (a) construtor, que recebe o total de questões (assuma que todas as questões possuem o mesmo valor e que o valor total da atividade é 100);

   (b) destrutor;

   (c) operação `void leNotas()`, que solicita ao usuário que digite a nota de cada questão (notas fora do intervalo válido devem ser lidas novamente);

   (d) operação `void relatorio()`, que calcula e imprime a nota final, além de imprimir mensagens indicando em quais questões as notas ficaram abaixo de 60%.

```
class Avaliacao
{
  private:
    int questoes;
    float valor;
    float *notas;

  public:
    Avaliacao(int n);
    ~Avaliacao();
    void leNotas();
    void relatorio();
};
```

4. (30 Pontos) Considere matrizes quadradas e simétricas de ordem n em que os elementos acima da diagonal principal são iguais à diferença entre os índices da respectiva coluna e da respectiva linha. Já os elementos abaixo da diagonal principal são iguais à diferença entre os índices da respectiva linha e da respectiva coluna. A matriz A é um exemplo desse tipo de matriz.

$$A = \begin{bmatrix} -5 & 1 & 2 & 3 & 4 \\ 1 & 6 & 1 & 2 & 3 \\ 2 & 1 & -8 & 1 & 2 \\ 3 & 2 & 1 & 7 & 1 \\ 4 & 3 & 2 & 1 & -9 \end{bmatrix}$$

O TAD `MatrizEspecial` representa esse tipo de matriz. Os valores devem ser armazenados no TAD `MatrizEspecial` numa representação linear com um único vetor (`vet`) e de modo que a quantidade de elementos armazenados seja mínima. Para esse TAD, desenvolver:

(a) Construtor, que recebe a ordem da matriz como parâmetro, e destrutor da classe.

(b) Operação `int detInd(int i, int j)`, que recebe a linha `i` e a coluna `j` como parâmetros e retorna: o índice correspondente no vetor `vet`, -1 se o índice da linha ou da coluna forem inválidos, e -2 se `i` e `j` não representam um valor no vetor `vet`.

(c) Operações `int get(int i, int j)` para acessar e `void set(int i, int j, int val)` para alterar valores da matriz. Se os índices `i` ou `j` forem inválidos, a operação `get` deve finalizar a execução do programa e a operação `set` deve imprimir uma mensagem de erro. Deve-se exibir uma mensagem de erro ao tentar atribuir um valor inválido para os elementos fora da diagonal principal.