Nome: _____ Matrícula: _____

Leia atentamente as instruções abaixo:

- Fazer o download do arquivo `Avaliacao1EDLab2-2017-3.zip` do site do curso. Descompactá-lo em um diretório de 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 marcador (//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, obrigatoriamente.
- 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.
- Antes de sair do laboratório, 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://172.18.40.97:8080/edlab2ufjf/`

Questões:

1. (20 Pontos) Desenvolva uma função que conte e retorne quantos elementos de um vetor `vet` de elementos inteiros, de tamanho `n`, são maiores do que um valor chave `ch` especificado. Essa função deve imprimir uma mensagem conforme exemplo abaixo para todos os elementos de `vet` que são maiores que `ch`. Em seguida, desenvolva uma função para alocar um vetor dinamicamente, copiar todos os elementos do vetor `vet` que são maiores que `ch` para esse novo vetor criado e, ao final, retornar esse vetor criado de forma dinâmica. Se o vetor não possuir nenhum elemento maior que `ch`, retornar `NULL`.

```
Protótipos:

int func1(int n, int *vet, int ch);
int* func2(int n, int *vet, int ch, int tam);

Exemplo de saída da func1:

posicao 0   valor 10   endereco 0x7fff9575c054
posicao 2   valor 33   endereco 0x7fff9575c058
...
```

2. (20 Pontos) Seja uma sequência de números, em que o n-ésimo valor é dado por:

$$e(n) = e(n-1) + 2 \times e(n-2) + 1.$$

Dado que $e(0) = 1$ e $e(1) = 2$, desenvolver uma função **recursiva** em C++ para calcular o n-ésimo elemento dessa sequência. Se o valor passado para `n` for menor do que zero, a função deve retornar -1. O protótipo da função é dado abaixo:

```
int sequencia(int n);
```

3. (30 Pontos) Abaixo encontram-se as classes que implementam os TADs `Equipe` e `Jogador`. Os dados armazenados para representar uma equipe são o tamanho da equipe e um vetor de jogadores (ponteiro para TAD `Jogador`). O construtor do TAD `Jogador` inicializa os atributos com valores aleatórios. Para o TAD `Equipe`, desenvolver:
   (a) construtor (que recebe o número de jogadores da equipe) e destrutor da classe;
   (b) operação `void imprimeEquipe()` que imprime as informações dos jogadores (número e pontos) da equipe, além do total de pontos de toda a equipe;
   (c) operação `int maiorPontuador()` que retorna o número do jogador que mais pontuou.

```
class Equipe
{
  private:
    int n;
    Jogador *jogadores;
  public:
    Equipe(int tam);
    ~Equipe();
    void imprimeEquipe();
    int maiorPontuador();
};
```

```
class Jogador
{
  private:
    int numero, pontos;
  public:
    Jogador(); ~Jogador();
    int getNumero();
    void setNumero(int n);
    int getPontos();
    void setPontos(int p);
};
```

4. (30 Pontos) Considere uma matriz `m` × `n` de elementos reais com `m` > 0 linhas e `n` > 0 colunas. Os elementos da primeira linha (`PL`) e última linha (`UL`) não são nulos. Também não são nulos os elementos da matriz da única linha central (`LC`). Portanto, a matriz tem 3 linhas de elementos não nulos. A representação vetorial, sem elementos nulos, da matriz é feita pela concatenação das linhas não nulas da seguinte forma: `vet` = [`PL LC UL`]. Se `m` for par, a linha central `LC` corresponde a linha central somando 1 a `m`. Ver exemplos para as matrizes `A(5×n)` e `A(6×n)` abaixo. Considere o TAD `Matriz3Linhas` definido a seguir, que armazena os elementos não nulos em um vetor `vet`. Para o TAD `Matriz3Linhas`, desenvolva:

$$A = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{0n-2} & a_{0n-1} \\ 0 & 0 & \dots & 0 & 0 \\ a_{20} & a_{21} & \dots & a_{2n-2} & a_{2n-1} \\ 0 & 0 & \dots & 0 & 0 \\ a_{40} & a_{41} & \dots & a_{4n-2} & a_{4n-1} \end{bmatrix} \qquad A = \begin{bmatrix} a_{00} & a_{01} & \dots & a_{0n-2} & a_{0n-1} \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ a_{30} & a_{31} & \dots & a_{3n-2} & a_{3n-1} \\ 0 & 0 & \dots & 0 & 0 \\ a_{50} & a_{51} & \dots & a_{5n-2} & a_{5n-1} \end{bmatrix}$$

   (a) construtor e destrutor da classe;
   (b) operação `int detInd(int i, int j)` que retorna -1 se os indices `i` ou `j` são inválidos; -2 se a linha `i` é uma linha correspondente a valores nulos; ou `k` para o índice de `vet` onde se encontra o elemento não nulo a_{ij} da matriz;
   (c) operação `float get(int i, int j)` que retorna o valor na posição `i,j` da matriz. Usar a função `detInd()`;
   (d) operação `void set(int i, int j, float val)` que atribui um valor na posição `i,j`. Usar a função `detInd()`. Exiba uma mensagem caso tente-se atribuir um valor não nulo em linhas de elementos nulos.

```
class Matriz3Linhas
{
  public:
    Matriz3Linhas(int mm, int nn);
    ~Matriz3Linhas();
    float get(int i, int j);
    void set(int i, int j, float val);
  private:
    int m, n;
    float *vet;
    int detInd(int i, int j);
};
```