UFJF / DCC Estrutura de Dados e Lab. de Programação II – TVC 1 17/04/2017

Nome: _____ Matrícula: _____

Leia atentamente as instruções abaixo:

- Fazer o download do arquivo `Avaliacao1EDLab2-2017-1.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 as turmas de ED e Lab. II.
- 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. (15 Pontos) Montar uma sequência de comandos que faça o que é solicitado. Os endereços de memória que aparecem são os que ocorreram numa dada execução e podem mudar a cada nova execução.
   (a) Declare uma variável `ab` do tipo ponteiro para inteiro;
   (b) Declare uma variável `c` do tipo inteiro;
   (c) Manipule `ab` e `c` seguindo as instruções da impressão e imprimindo o que segue.

```
Atribuir 29 a c.
Endereco de memoria de c: 0x7ffcc3ad291c
Valor de c: 29

Atribuir o endereco de memoria de c a ab.
Valor de ab: 0x7ffcc3ad291c
Endereco de memoria de ab: 0x8eedd4be380b
Conteudo apontado por ab: 29
```

```
Atribuir 34 a c.
Valor de ab: 0x7ffcc3ad291c
Endereco de memoria de ab: 0x8eedd4be380b
Conteudo apontado por ab: 34

Atribuir 7 ao valor apontado por ab.
Endereco de memoria de c: 0x7ffcc3ad291c
Valor de c: 7
```

2. (15 Pontos) O código que segue deveria imprimir todas as combinações de uma sequência de caracteres. Entretanto, há algum problema no código pois apenas a própria sequência está sendo impressa. Deve-se corrigir o código para que as permutações de "abcde" sejam impressas.

```
void funcao1(char a, char b);
void funcao2(char *s, int l, int r);

int main() {
  char str[] = "abcde";
  funcao2(str, 0, 4);
}

void funcao1(char a, char b) {
  char c = a;
  a = b;
  b = c;
}
```

```
void funcao2(char *s, int l, int r) {
  if (l == r)
    cout << s << endl;
  else
  {
    for(int i = l; i <= r; i++)
    {
      funcao1(s[l], s[i]);
      funcao2(s, l+1, r);
      funcao1(s[l], s[i]);
    }
  }
}
```

3. (15 Pontos) Desenvolver a função **recursiva** `soma` em C++ para, dados `n` e `x`, calcular e retornar o seguinte somatório:

$$\text{soma}(x,n) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots + (-1)^{n-1}\left(\frac{x^n}{n}\right) = \sum_{i=1}^{n}(-1)^{i-1}\left(\frac{x^i}{i}\right)$$

```
float soma(float x, int n);
```

4. (25 Pontos) Abaixo encontram-se as classes que implementam os TADs Triângulo e Ponto. Os dados armazenados para representar um triângulo são os seus 3 pontos (três ponteiros para um TAD Ponto, com coordenadas x e y). Para o TAD Triângulo, desenvolva:
   (a) construtor e destrutor da classe; `//atenção às construções de triângulos inválidos`
   (b) operação `int tipo()` que retorna qual o tipo do triângulo (equilátero - 0, isósceles - 1 ou escaleno - 2);
   (c) operação `float areaEq()` que retorna a área de um triângulo equilátero ($a = l^2 \times \sqrt{3} \div 4$). Se o triângulo não for equilátero, retornar 0.

```
class Triangulo {
 public:
   Triangulo(float x1, float y1,
             float x2, float y2,
             float x3, float y3);
   ~Triangulo();
   int tipo();
   float areaEq();
 private:
   Ponto *p1, *p2, *p3;
};
```

```
class Ponto {
 public:
   Ponto(float xx, float yy);
   ~Ponto();
   float getX(); float getY();
   void setX(float xx);
   void setY(float yy);
   float distancia(Ponto *p2);
 private:
   float x, y;
};
```

5. (30 Pontos) Considere uma matriz quadrada de $n \times n$ elementos inteiros com uma estrutura conforme a matriz abaixo, em que somente a diagonal secundária e as diagonais imediatamente acima (superdiagonal) e abaixo (subdiagonal) da secundária são diferentes de zero. Considere o TAD `MatrizTriDiagSec` definido a seguir, que armazena os elementos não nulos em 3 vetores (`diagSec`, `superDiagSec` e `subDiagSec`). Para o TAD `MatrizTriDiagSec`, desenvolva:

$$A = \begin{bmatrix} 0 & 0 & 0 & a_{03} & a_{04} \\ 0 & 0 & a_{12} & a_{13} & a_{14} \\ 0 & a_{21} & a_{22} & a_{23} & 0 \\ a_{30} & a_{31} & a_{32} & 0 & 0 \\ a_{40} & a_{41} & 0 & 0 & 0 \end{bmatrix}$$

   (a) construtor e destrutor da classe;
   (b) operação `float consulta(int i, int j)` que retorna o valor na posição `i,j` da matriz;
   (c) operação `void atribui(int i, int j, float val)` que atribui um valor na posição `i,j`. Exiba uma mensagem caso tente-se atribuir um valor que descaracterize a matriz.

```
class MatrizTriDiagSec
{
  public:                                  private:
    MatrizTriDiagSec(int dim);               int n;
    ~MatrizTriDiagSec();                     float *diagSec,*superDiagSec,*subDiagSec;
    float consulta(int i, int j);            bool verifica(int i, int j);
    void atribui(int i, int j, float val);
};
```