UNIVERSIDADE FUMEC/FACE
Faculdade de Ciências Empresariais
Curso: Ciência da Computação – BH
Disciplina: ED-I – Professor: Ricardo Terra

INFORMAÇÕES SOBRE ESTA ATIVIDADE:

- Esta atividade valerá 2,0 pontos (*1,0 ponto de trabalho mais 1,0 ponto de AAI*).
- A atividade deve se iniciar no laboratório e ser entregue pelo SINEF **até dia 30/out/2012**.
- Deve ser entregue um arquivo compactado chamado **`Ativ06.zip`** com um arquivo chamado **`grupo.txt`** contendo o nome dos integrantes e os arquivos **`c`** cujos nomes devem seguir o seguinte padrão: **`Ativ06Ex01.c, Ativ06Ex02.c,...`** **Isto será avaliado!**
- Apenas **um** do grupo deverá entregar. **Isto também será avaliado!**
- Atividade não serão recebidas/pontuadas **após o prazo de entrega**. Portanto, fique atento ao prazo.
- **Cópias** (total ou parcial) serão penalizadas com **nota zero**.
- A atividade é em grupo de, no máximo, **dois** alunos.

Atividade 06 - Lista de Exercícios – Ponteiros – Parte 2

Dica: É boa prática a utilização de protótipos.

1. Observe o trecho de chamada do código. A função *copia* realiza a cópia dos valores de um arranjo de inteiros para um outro arranjo de mesmo tamanho.

```
int main (...){
    int vA[] = {1,2,3,4,5};
    int vB[5];
    copia(vA, vB, sizeof(vA)/sizeof(int));
    return 0;
}
```

   a. Crie a função **`copia`** com a seguinte assinatura e realizando a cópia utilizando indexação.

```
void copiaComIndexacao(int vA[], int vB[], int tam)
```

   b. Crie a função **`copia`** com a seguinte assinatura e realizando a cópia **sem utilizar indexação**.

```
void copiaSemIndexacao (int *vA, int *vB, int tam)
```

2. Observe o trecho de chamada do código. A função **`troca`** realiza a troca dos valores de um arranjo de **`double`** para um outro arranjo de mesmo tamanho.

```
int main (...){
    double vA[] = {0.1 , 0.2 , 0.3 , 0.4 , 0.5};
    double vB[] = {1.1 , 2.2 , 3.3 , 4.4 , 5.5};
    troca(vA, vB, 5);
}
```

3. Crie a função **`troca`** com a seguinte assinatura e realizando as trocas **sem utilizar indexação**.

```
void troca (double *vA, double *vB, int tam)
```

4. Crie um programa C ANSI que tenha uma função **`inverte`** que receba uma *string* e a inverta. Deve ser utilizada a seguinte assinatura e **nenhuma indexação deve ser feita**: `void inverte (char *str)`

5. Dada a estrutura abaixo:

```
typedef struct {
    short int dia;
    short int mes;
    int ano;
} data;
```

   Criar um programa C ANSI que realize a leitura de uma data através da função **`leData`** (assinatura abaixo) e depois realize a impressão da data no formato *dia/mês/ano* através da função **`imprimeData`** (assinatura abaixo). Quando possível, utilizar o operador seta (**->**).

```
void leData (data *d)      e    void imprimeData (data *d)
```

6. Dada a estrutura abaixo:

```
typedef struct{
    char logradouro[60];
    int numero;
    char bairro[60];
    char cidade[60];
    char uf[3];
    long int cep;
}endereco;

typedef struct{
    int codigo;
    char cpf[12];
    char nome[60];
    endereco end;
}cliente;
```

Criar um programa C ANSI que realize a leitura de um cliente através da função **`leCliente`** (assinatura abaixo) e depois realize a impressão das informações do cliente através da função **`imprimeCliente`** (assinatura abaixo). Quando possível, utilizar o operador seta (**->**).

`void leCliente (cliente *c)` e `void imprimeCliente (cliente *c)`

7. Dada a estrutura abaixo:

```
typedef struct {
    int valor;
    short int ativo;
} nodo;
```

Criar um programa C ANSI que crie um arranjo de 10 nodos ativos (`ativo=1`) com valores aleatórios e chame uma função **`desativa`** (assinatura abaixo) que desative todos os nodos que possuam aquele valor **sem utilizar nenhuma indexação**. Finalmente, imprima todos os nodos ativos para comprovar o funcionamento.

```
void desativa(nodo *v, int valor, int tam)
```

Por exemplo, foi criado o seguinte arranjo **`nodos`**:

`{[10,1], [6,1], [4,1], [3,1], [7,1], [8,1], [8,1], [6,1], [9,1], [1,1]}`

Ao chamar **`desativa(nodos, 6, sizeof(nodos)/sizeof(nodo))`** o arranjo ficará como a seguir:

`{[10,1], **[6,0]**, [4,1], [3,1], [7,1], [8,1], [8,1], **[6,0]**, [9,1], [1,1]}`

8. [LOGICA] Crie um programa C ANSI que declare um *string* (arranjo de caracteres) de pelo menos 2 posições e imprima cada posição dele **sem utilizar indexação** (isto é, aritmética de ponteiro).
Além disso, você deve sempre imprimir o primeiro e último caracteres. No entanto, os caracteres do meio só serão impressos se a média da soma do código ASCII do anterior e do posterior for menor que o código do caractere ASCII a ser impresso.

Exemplos (*string* de entrada → saída do programa):

`"ANA"`	→ `ANA`	`"CHOCOLATE"`	→ `COOLTE`
`"CASA"`	→ `CSA`	`"RICARDO TERRA"`	→ `RROTRRA`

9. [LOGICA] Crie um programa C ANSI que declare um arranjo de 10 inteiros. Sem utilizar indexação, aponte um ponteiro para a 1ª posição do arranjo. Basicamente, você movimentará o ponteiro de acordo com os valores encontrados. Quando achar um 0, a soma dos valores até então deverá ser impressa.

Exemplos:
`[**7,** 1, 4, 5, 0, **0,** 7, **-2,** 13, 8]` → `5 (isto é, 7 + (-2))`
`[**3, 5, -1, 4,** 1, 18, **0, -5,** 2, 8]` → `6 (isto é, 3 + 4 + (-5) + (-1) + 5)`

Tratar ciclo:
`[**8, 1, 0,** 4, 7, 18, 0, -5, **3,** 8]` → `12 (isto é, 8 + 3 + 1)`

Tratar *loop* eterno:
`[**1, 6,** 3, 7, 7, 18, 0, **-7,** 3, 8]` → `VALOR 0 NÃO ACESSÍVEL`