UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA

Introdução às Técnicas de Programação — IMD0012 — 29 de setembro de 2017
◁ Exercícios - parte 9 ▷

Observação:

- Siga rigorosamente a especificação da função
- Nenhuma string nos arquivos ultrapassa 50 caracteres
- Durante a correção, o professor conferirá se a função foi utilizada para resolver a questão
- Em algumas questões, citar um ponteiro como um vetor é uma forma curta de dizer que no endereço para o qual esse vetor aponta há um vetor
- Não esqueça de liberar a memória que foi alocada dinamicamente ao terminar de usá-la.

1. ▷ Crie uma função que retorna o maior entre dois inteiros:

```
int maior2(int a, int b);
```

Sem usar condicionais, apenas utilizando a função maior2, implemente a função que retorna o maior entre três inteiros:

```
int maior3(int a, int b, int c);
```

A função *main* deve ler do usuário 3 inteiros e escrever na tela o maior entre os 3 utilizando a função maior3.

2. ▷ Crie uma função que retorna a quantidade de divisores de um número **x** passado como parâmetro:

```
int qtdDivisores(int x);
```

Sem usar repetições, apenas utilizando a função acima, crie uma função que retorna se um número **x** é primo ou não (valor lógico).

```
int ehPrimo(int x);
```

A função *main* deve ler do usuário um inteiro (assuma que o usuário sempre digita um valor maior que 0) e escrever na tela 1 se o número for primo e 0 caso contrário.

3. ▷ Crie uma função que armazena a partir da derreferenciação dos ponteiros **dx** e **dy** a diferença entre os pontos no plano cartesiano (px, py) e (qx, qy) em relação ao eixo x e y, respectivamente:

$$dx = qx - px$$

$$dy = qy - py$$

```
void diferenca(int px, int py, int qx, int qy, int *dx, int *dy);
```

A função *main* deve ler do usuário 4 inteiros representando os 2 pontos e escrever na tela a diferença em cada um dos eixos.

Nota do professor: como em C não é possível retornar dois valores explicitamente, nessa questão você precisa utilizar ponteiros para recuperar os valores (dx, dy) na função main. Há a possibilidade de retornar uma **struct** com os dois, mas esse é um assunto a ser visto ainda.

4. ▷ Crie uma função que **modifica** uma string substituindo todos os caracteres **x** pelo caractere **y**:

```
void substituir(char *s, char x, char y);
```

Por exemplo, ao chamar substituir(palavra, 'a', 'e'), sendo palavra "lata" , torna-se "lete" .

A função *main* deve ler do usuário uma string, ler o caractere será substituído, ler o caractere que o substituirá e, em seguida, escrever na tela a string resultante. A string é finalizada com um
n, seguido de **exatamente** o primeiro caractere, espaço e o segundo caractere (sem mais, nem menos).

Nota do professor: há duas observações ao chamar a função substituir("lata", 'a', 'e'): primeiro "lata"está na memória, mas não pode ser modificada (ocasiona falha de segmentação se tentar) e, segundo, mesmo que fosse possível alterar, teria que armazenar primeiramente o endereço da string "lata"para depois poder escrevê-la. O ideal portanto é na função main declarar uma string.

5. ▷ Crie uma função que retorna uma nova string igual à passada como parâmetro mas substituindo todos os caracteres **x** pelo caractere **y**:

```
char * substituir(char *s, char x, char y);
```

Por exemplo, ao chamar substituir(palavra, 'a', 'e'), sendo palavra "lata" , retorna "lete" .

A função *main* deve ler do usuário uma string, ler o caractere será substituído, ler o caractere que o substituirá e, em seguida, escrever na tela a string resultante, seguida da original. A string é finalizada com um
n, seguido de **exatamente** o primeiro caractere, espaço e o segundo caractere (sem mais, nem menos).

Nota do professor: Aqui ao contrário da questão anterior, ao chamar a função substituir("lata", 'a', 'e'), esta retorna uma nova string e não altera propriamente "lata".

6. ▷ Crie uma função para ordenar um vetor **v** de **n** inteiros em ordem **decrescente**:

```
void ordenar(int *v, int n);
```

A função *main* deve ler do usuário um inteiro n, **alocar dinamicamente** um vetor de n inteiros, ler os **n** inteiros do vetor, chamar a função e escrever na tela os valores do vetor.

7. ▷ Crie uma função para retornar o endereço base de um vetor de **t** inteiros alocado dinamicamente cujo conteúdo é lido da **entrada padrão**. Por exemplo, ao chamar lerVetorInteiros(5) a função deve retornar o endereço base de um vetor de 5 inteiros, cujos valores já são lidos através da **entrada padrão** dentro da própria função.

```
int * lerVetorInteiros(int t);
```

Crie uma função para retornar o endereço base de um vetor de inteiros alocado dinamicamente cujo conteúdo é a concatenação dos números contidos nos vetores **u** (tamanho **m**) e **v** (tamanho **n**):

```
int * juntarVetores(int *u, int *v, int m, int n);
```

A função *main*:

```
int main() {
    int *v1, *v2, tam1, tam2, *v3;
    scanf("%d", &tam1);
    v1 = lerVetorInteiros(tam1);
    scanf("%d", &tam2);
    v2 = lerVetorInteiros(tam2);

    //armazenar em v3 o resultado da juncao dos vetores v1 e v2
    //escrever na tela os numeros de v3
    return 0;
}
```