PUC-Minas - Ciência da Computação
AED1 – Lista de Exercícios para Recuperação 01

Tema: Recuperação 01
Atividade: Conceitos básicos

INSTRUÇÕES:
- Esses exercícios são extras, opcionais.
  O objetivo é rever aplicações de conceitos essenciais.

- Desenvolver métodos para cada um dos enunciados abaixo.
- Cada método deverá conter, em seu cabeçalho, como comentário
  (/** e */), a documentação essencial:
  nome e matrícula,
  identificação, objetivo, parâmetros e condições especiais,
  se houver, e relatório de testes
  (exemplos de valores usados e condições testadas).
- Um programa com todos os métodos deverá ser usado para testes.

SUGESTÃO: Montar um menu para a escolha do método a ser testado
(ver modelo em Lista00).
Testes deverão ser realizados e os valores usados deverão
ser guardados no final do programa como comentários (/* e */).
As funções não deverão mostrar as respostas ou os valores calculados.

01.) DEFINIR funções e procedimento para testes (exercicio01) para:
- ler um valor inteiro (n) do teclado para indicar uma quantidade;
- ler outros tantos valores inteiros (x) do teclado, um por vez, até atingir a quantidade;
- dizer para cada valor lido se há mais divisores ímpares do que pares.

DICA: Usar apenas alternativas simples.
O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
n = 5
x = { 22, 121, 440, 581, 814 }
soma_impares = soma_divisores_impares (valor);
soma_pares = soma_divisores_pares (valor);

02.) DEFINIR função e procedimento para testes (exercicio02) para:
- ler um valor inteiro (n) do teclado para indicar uma quantidade;
- ler outros tantos valores inteiros (x) do teclado, um por vez, até atingir a quantidade;
- dizer para cada valor lido se é primo, ou não, verificado por função.

DICA: Usar alternativas dupla.
O resultado NÃO deverá ser mostrado dentro da função.
NÃO usar **break** ou **continue**.
Usar apenas 01 retorno.
O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
n = 5
x = { -293, -107, 153, 199, 811 }
resposta = primo (valor);

03.) DEFINIR funções e procedimento para testes (exercicio03) para:
- ler uma cadeia de caracteres do teclado;
- dizer se cada símbolo é um operador lógico (&,|,!), aritmético (+,-,*,/,%), relacional (>,<,=), separadores (espaço, ponto, vírgula, ponto-e-vírgula, dois-pontos, sublinha) ou outro símbolo qualquer.

DICA: Definir e usar funções para identificar cada um desses tipos de símbolo.
O resultado NÃO deverá ser mostrado dentro da função.
Usar **switch** para escolhas.
Usar apenas 01 retorno.
O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
"for(x=0;x<y&&y>=0;x=x+1)"
resposta = testar_simbolo (valor);

04.) DEFINIR funções e procedimento para testes (exercicio04) para:
- ler um cadeia de caracteres (s) do teclado;
- testar cada símbolo na cadeia, um por vez, e contar separadamente quantos desses são letras maiúsculas, minúsculas, dígitos e operadores (lógicos, aritméticos e relacionais) e separadores.

DICA: Definir e reutilizar funções já definidas para identificar cada tipo de símbolo (ver acima).
Os resultados NÃO deverão ser mostrados dentro das funções.
Usar **switch** para escolhas e somas.
Usar apenas 01 retorno.

Exemplos:
"for(X=0;X<y&&y>=0;X=X+1)"
resposta = testar_tipo (valor);

05.) DEFINIR função e procedimento para testes (exercicio05) para:
- ler um valor inteiro (n) do teclado;
- ler outros (n) valores reais (x) do teclado, um por vez;
- calcular e mostrar a média dos valores menores que -27.25;
  a média dos que estão entre -27.2 e 91.25, inclusive esses;
  e a média dos maiores que 91.25. Dizer qual a menor média de todas.
  DICA: Definir e usar função para calcular o menor dentre três valores.
  O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
n = 10
x = { -15.25, -12.50, 0.0, 6.75, 20.50, 50.0, 70.25, 85.25, 92.50, 98.75 }
resposta1 = no_intervalo (inferior, superior, valor);

06.) DEFINIR função e um procedimento para testes (exercicio06) para:
- ler dois valores inteiros (a) e (b) do teclado,
- ler outros valores inteiros do teclado, um por vez,
  até que o último valor seja igual a (-1).
- para cada valor lido que estiver no intervalo aberto]a:b[;
  calcular e mostrar a soma dos cubos dos inversos ($1/x^3$) dos ímpares não múltiplos de 3.
  DICA: Evitar divisão por zero mediante uso de alternativa dupla.
  Definir e usar funções para testar o pertencimento ao intervalo,
  e para calcular a soma.
  O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
a = 20
b = 60
x = { -20, -11, 0, 1, 10, 21, 36, 51, 70, 80, -1 }
resposta = soma_cubos (a, b);

07.) DEFINIR função e um procedimento para testes (exercicio07) para:
- ler dois valores reais (a) e (b) do teclado,
- ler outros valores reais do teclado, um por vez,
  até que o último valor seja igual a zero,
  e não considerá-lo para os cálculos.
- contar quantos desses valores estão dentro ou fora
  do intervalo definido pelo intervalo aberto]a:b[;
- para os valores fora do intervalo, calcular separadamente
  as porcentagens dos que estiverem acima e abaixo do intervalo, nessa ordem.

DICA: Evitar divisão por zero mediante uso de alternativas duplas.
Reutilizar a função para testar o pertencimento (ou não) ao intervalo.
O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
a = 20
b = 60
x = { -20, -10, 1, 10, 20, 30, 60, 70, 80, 0 }
resposta = no_intervalo (a, b, valor);

08.) DEFINIR função e um procedimento para testes (exercicio08) para:
- ler três valores reais (x,y,z) do teclado, um por vez;
- dizer se esses valores estão em ordem crescente,
  decrescente ou em nenhuma dessas ordens e, nesse caso, o menor e o maior.

DICA: Usar testes com maior número de comparações usando conectivos lógicos (&&, || e !).
Definir funções para testar as ordens crescente e decrescente.
O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
x = 10, y = 20, z = 30
x = 30, y = 20, z = 10
x = 10, y = 10, z = 10
resposta1 = crescente (x, y, z);
resposta2 = decrescente (x, y, z);
resposta3 = maior (x, y, z);
resposta4 = menor (x, y, z);

09.) DEFINIR função e um procedimento para testes (exercicio09) para:
- ler três caracteres (x,y,z) do teclado, um por vez;
- dizer se esses valores estão em ordem crescente,
  decrescente ou em nenhuma dessas ordens.
  DICA: Usar uma função lógica para comparar os códigos inteiros de cada caractere,
  e dizer se o segundo sucede o primeiro em ordem alfabética.
  Definir funções para testar as ordens alfabéticas crescente e decrescente.
  O resultado NÃO deverá ser mostrado dentro da função.

Exemplos:
x = '1', y = '2', z = '3'
x = '3', y = '2', z = '1'
x = '1', y = '1', z = '1'
resposta1 = crescente (x, y, z);
resposta2 = decrescente (x, y, z);
resposta3 = sucessor (x, y);

10.) DEFINIR função e um procedimento para testes (exercicio10) para:
- ler três cadeias de caracteres (x,y,z) do teclado, uma por vez;
- dizer se essas cadeias estão em ordem alfabética decrescente,
  decrescente ou em nenhuma dessas ordens e, nesse caso, a menor e a maior.
  DICA: Usar strcmp( ) da biblioteca <string.h> e comparar o resultado com zero.
  A função para a comparação poderá retornar um dentre
  os seguintes resultados:
  O resultado NÃO deverá ser mostrado dentro da função.

negativo, se a primeira cadeia preceder a segunda (resultado menor que zero);
nulo , se as cadeias forem iguais (resultado igual a zero);
positivo , se a primeira cadeia suceder a segunda (resultado maior que zero).

Exemplo:

```
char s1 [ ] = “abc”;
char s2 [ ] = “ABC”;
if ( strcmp ( s1, s2 ) != 0 )
{
   printf ( "%s\n", “diferentes” );
} // end if
```