Atividade prática sobre funções e ponteiros

1) **[Adaptação da questão 35 do Poscomp 2006]** Que valores são impressos quando o seguinte algoritmo, escrito em C, é executado?

```
#include <stdio.h>
int a, b;
void Mist( int x, int *y )
{  x = *y + a + 1;
   *y = x + b + 1;
}
int main()
{   a = 1; b = 0;
    Mist( a, &b );
    printf( "%d %d", a, b );
}
```

a) 3 1

b) 1 7

c) 3 5

d) 4 7

e) 1 3

2) **[Questão 39 do Poscomp 2011]** Considere a função desenvolvida na Linguagem C, a seguir.

```
char *Teste (char *s1, const char *s2)
{
      char *aux = s1;
      while (*s1) s1++;
      for (;(*s1 = *s2) != '\0'; s1++, s2++);
      return aux;
}
```

O seu objetivo é:

a) Copiar o conteúdo da região de memória referenciada pelo identificador `s1` para a região de memória referenciada pelo identificador `s2`.

b) Atribuir o valor '\0' para todas as posições de memória entre o endereço referenciado pelo identificador `s1` até a região de memória referenciada pelo identificador `s2`.

c) Comparar o conteúdo de memória que se inicia na posição referenciada pelo identificador `s1` e ir até a ocorrência de um valor '\0' com o conteúdo da região de memória referenciada pelo identificador `s2`.

d) Substituir os elementos armazenados na região de memória referenciada pelo identificador `s1` pelos elementos armazenados na região de memória referenciada pelo identificador `s2`.

e) Copiar os elementos contidos na região de memória referenciada pelo identificador `s2` após os elementos armazenados na região de memória referenciada pelo identificador `s1`.

3) Considere a definição de variáveis a seguir e assinale com V as afirmações verdadeiras e com F as falsas.

```
(…)
010 struct ponto {float x; float y;};
011 struct ponto v1;
012 struct ponto *p1;
013 float distancia, *f;
(…)
```

( ) Existem 3 variáveis declaradas no trecho de código apresentado: `v1`, `ponto`, `distancia`.

( ) Tanto a variável `distancia` como a variável `v1` são de tipos primitivos da linguagem C.

( ) A variável `v1` pode conter as coordenadas de um ponto no plano bidimensional.

( ) Considerando os tipos de dados da **`struct`** `ponto`, podemos concluir que a variável `v1` deve possuir exatamente 5 bytes nas implementações padrão de 32 bits da linguagem C.

( ) A instrução **`sizeof`**`(`**`struct`** `ponto *)` poderia responder adequadamente a questão indicada no item anterior.

4) Considere o programa corretamente codificado em C apresentado a seguir e faça o que se pede.

```
001 int main(void)
002 {   float a, b, c, area, semip;
003     while (1)
004     {    printf("Informe os lados do triangulo: ");
005          scanf("%f %f %f", &a, &b, &c);
006          if (a == 0.0 || b == 0.0 || c == 0.0)
007               break;
008          if (a < b + c && a > abs(b - c))
009          {    semip = (a + b + c) / 2;
010               area = pow(semip * (semip - a) * (semip - b)
* (semip - c), 0.5);
011               printf("\nA area eh: %f\n\n", area);
012          }
013          else
014               printf("Nao eh um triangulo\n\n");
015     }
016     return 0;
017 }
```

Reescreva o programa adaptando-o para:

a) usar uma *struct* para armazenar as medidas dos 3 lados do triângulo;

b) usar uma função para computar a área do triângulo, conforme o cálculo que é realizado nas linhas `009` e `010` do programa;

c) usar uma função para verificar se as medidas fornecidas pelo usuário podem ou não formar um triângulo. A rotina deverá retornar 0 se as medidas não correspondem a um triângulo, 1 for um triângulo escaleno, 2 se for isósceles e 3 se for equilátero. Considere que para um conjunto de medidas a, b e c poder formar um triângulo, a condição verificada na linha `008` do programa deve ser verdadeira.

5) Uma importante medida estatística é a variância, usada para avaliar o grau de dispersão de um conjunto de números em relação à sua média. A fórmula da variância é dada por

$$V = \frac{\Sigma (x - m)^2}{}$$

$$\frac{}{N}$$

onde N é a quantidade de elementos e m é a média aritmética simples dos valores, que é dada por

$$m = \frac{\Sigma x}{N}$$

Escreva um programa que recebe inicialmente um valor de N (um inteiro entre 1 e 100), indicando a quantidade de números a serem considerados. Em seguida o programa deverá receber e armazenar em um vetor N números reais de precisão simples (tipo `float` da linguagem C), calculando e imprimindo (com 4 casas depois da vírgula) a média e a variância desses valores conforme as fórmulas anteriormente apresentadas, também utilizando números reais de precisão simples. Usar uma função para calcular a média dos valores do vetor e outra função para calcular a variância. Imprimir os resultados apenas na rotina principal (a rotina `main()`).

Exemplo de entrada:

```
12
1.0  3.5  0.5  8.9  19.2  88.91  1.23  8.42  7.98  -12.52
128.39  -200.57
```

Saída esperada para o exemplo de entrada:

```
Media: 4.5783          Variancia: 5426.6636
```

6) Faça um programa que recebe uma palavra digitada pelo usuário, contendo até 20 caracteres, e a imprime em minúsculas. Utilize um função para converter a string para minúsculas, conforme o exemplo contido no arquivo 40_ED-Subrotinas&Parametros.pdf.

7) Implemente o programa C abaixo, definindo também a rotina `Cripto`, conforme as especificações apresentadas a seguir:

```
#include <stdio.h>
int main(void)
{  char texto1[9], texto2[9], texto3[9];

   printf("Informe um texto de até 8 caracteres:");
   scanf("%[^\n]", texto1);

   Cripto(texto1, texto2);
   Cripto(texto2, texto3);
   printf("O    texto    criptografado    eh:    \n%s\n    e
descriptografado eh: \n%s\n", texto2, texto3);

   return 0;
}
```

Rotina: Cripto
Interface da rotina:
Recebe 2 parâmetros: Variável contendo a string a ser (des)criptografada e variável onde o resultado deverá ser armazenado
Comportamento:

Com base no conteúdo de duas strings de controle definidas dentro da subrotina, esta rotina gera a criptografia de um dado. Sua lógica interna consiste em obter um a um os caracteres da string original e, após pesquisá-los nas cadeias de controle, substituí-los conforme o seguinte princípio:

- Se o caracter existir na primeira string de controle, colocar em seu lugar o caracter equivalente na segunda string de controle;
- Senão, se o caracter existir na segunda string de controle, colocar em seu lugar o caracter equivalente na primeira string de controle;
- Senão, se o caracter não existir em nenhuma das duas strings de controle, deixá-lo inalterado.

Dicas:

Montar as duas strings de controle com o mesmo tamanho, mas sem nenhum tipo de repetição de caracteres. Considerar diferentes as versões maiúscula e minúscula de uma mesma letra. Utilizar, além das letras, também números e caracteres especiais.

8) Adapte o programa pRA_2.c, disponível na pasta de arquivos da equipe LP no Teams, para que utilize uma função para determinar o novo RA.