1ª Lista de Exercícios para Nota

1) Implemente uma função que calcule a área da superfície e o volume de uma esfera de raio R. Essa função deve obedecer ao protótipo:

void calc_esfera(float R, float *area, float *volume)

A área da superfície e o volume são dados, respectivamente, por:

$$A = 4 * \pi * R^2 \qquad V = 4/3 * \pi * R^3$$

2) Escreva uma função que aceita como parâmetro um vetor de inteiros com N valores, determina o menor elemento do vetor e o número de vezes que este elemento ocorreu no vetor. Por exemplo, para um vetor com os seguintes elementos: 5, 2, 15, 3, 7, 2, 8, 6, 2, a função deve retorna para o programa que a chamou o valor 2 e o número 3 (indicando que o número 2 ocorreu 3 vezes). A função deve obedecer ao protótipo:

void menor(int vet[], int *menor, int *vezes)

3) Considere um cadastro de produtos de um estoque, com as seguintes informações para cada produto:

• Código de identificação do produto: representado por um valor inteiro
• Nome do produto: com até 50 caracteres
• Quantidade disponível no estoque: representado por um número inteiro
• Preço de venda: representado por um valor real

a) Defina uma estrutura, denominada produto, que tenha os campos apropriados para guardar as informações de um produto
b) Crie um vetor (alocação dinâmica) de N produtos (N e um valor fornecido pelo usuário) e peça ao usuário para entrar com as informações de cada produto
c) Encontre o produto com o maior preço de venda
d) Encontre o produto com a maior quantidade disponível no estoque

4) Faça uma função recursiva para calcular os números de Pell. Os números de Pell são definidos pela seguinte recursão:

p(n) = 0 se n = 0
p(n) = 1 se n = 1
p(n) = 2p(n-1) + p(n-2) se n > 1

5) Faça uma função recursiva que calcule e retorne o N-ésimo termo da sequência Fibonacci. Alguns números desta sequência são: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

6) Considere uma estrutura de lista ligada:

```
typedef struct no {
        int info;
        struct no *prox;
}lista;
```

a) implemente uma função para retirar um elemento de uma posição determinada. Adote como primeiro valor a posição 0 (zero).

Cabeçalho: **lista *retiraDaPosicao(lista *no, int pos)**

b) implemente uma função para comparar o tamanho das listas encadeadas.

Cabeçalho**: int comp_tamanho(lista *l1, lista *l2)**.

Se as listas tiverem o mesmo tamanho retorna 0, se a lista l1 for menor retorna -1, caso contrário, retorna 1.