**Os algoritmos pedidos nos itens 1, 3 e 4 podem ser escritos em C “puro”, C++, C# ou até mesmo em pseudo-código.**

**O item 2 não necessita de algoritmo; 5 e 6 são em SQL.**

**1)** Escreva uma função que receba como parâmetros o primeiro elemento de uma progressão aritmética, a razão dessa progressão, e um número n, e devolva o n-esimo elemento dela.

Não é necessário usar fórmulas prontas, basta lembrar que uma PA começa em um certo valor e vai sendo somado um valor fixo (chamado razão)

Ex1: Prim = 3, Razão = 2: 3, 5, 7, 9, ....(o 4º elemento aqui é o 9)

Funcao\_PA (3, 2, 4) deve devolver 9

Ex1: Prim = 10, Razão = 5: 10, 15, 20, 25, 30, 35, 40, ....

Funcao\_PA (10, 5, 6) deve devolver 35

**2)** Para A = True, B = True e C = False, qual o resultado da avaliação das seguintes expressões:

a) (A **ou** B) **e** (A **e** C)

b) **não** B ou (A **ou** C)

**3)** Escreva uma função chamada Parece, que recebe como parâmetro dois vetores A e B, e mais o número de elementos de cada um. Essa função retorna True se os vetores forem “parecidos” ou False, caso contrário. O critério para ser “parecido” é que pelo menos 90% dos elementos de A estejam em B e 80% dos elementos de B estejam em A.

**4)** Escreva um algoritmo que tenha como entrada dois vetores, chamados Vetor1 e Vetor2, já populados (preenchidos) com números positivos em ordem crescente. O número de elementos dos vetores, são dados, respectivamente, pelas variáveis nVetor1 e nVetor2.

A rotina deve preencher um terceiro vetor, chamado Vetor3, preferencialmente de forma ordenada, só com os números que estejam em ambos os vetores de entrada.

Na linguagem C, as declarações das variáveis seriam algo como:

long Vetor1 [100];

long Vetor2 [100];

long Vetor3 [100]

long nVetor1;

long nVetor2;

long nVetor3;

Abaixo, segue um exemplo:

Dados entrada (já populados)

Vetor1 = [3, 4, 6, 8]

Vetor2 = [1, 2, 3, 4, 5, 9]

nVetor1 = 4

nVetor2 = 6

Saída esperada da rotina para este exemplo:

Vetor3 = [3, 4]

nVetor3 = 2

**5)** Criar uma estrutura de banco RELACIONAL a seguir:

- Clientes (campos Nome e Endereço)

- Produtos (Campos Código e Descrição)

- Vendas

Cada Venda representa um produto vendido a um cliente. Possui campos Quantidade (representa quantos itens do produto), Preço total da Venda, e Data da Venda

- Detalhes:

Um Produto pode ser vendido várias vezes e a vários clientes

Um cliente pode comprar várias vezes e vários produtos

**6)** Baseado na estrutura criada no item **(6)**, montar escrever uma SQL simples para obter:

a) Quantidade total de venda de cada Produto (exibir Código do produto)

b) Quantidade total de compra de cada cliente (exibir Nome do cliente)

c) Valor total vendido a cada dia

d) Quais Produtos não foram vendidos nunca (exibir Código do produto)

e) Quais clientes nunca compraram (exibir Nome do cliente)

f) Quantos produtos diferentes cada cliente comprou, independente de quantidade de venda, preço e data (exibir Nome do Cliente)