Recursividade

1. Uma soma pode ser definida recursivamente conforme abaixo:

$$\sum_{k=m}^{n} k = \begin{cases} m & \text{se } n = m \\ m + \sum_{k=m+1}^{n} k & \text{se } n > m \end{cases}$$

Implemente uma classe `Soma` que contenha uma **função recursiva** que receba dois parâmetros m e n e retorna o valor da soma conforme a definição acima. Em seguida, inclua no seu `main.cpp` uma instrução para testar essa função. O programa deve ler do usuário os valores de m e n, chamar a função e imprimir na tela o valor retornado pela função.

Exemplo de execução do programa:

Digite m: **1**
Digite n: **4**
10

2. Implemente uma classe `power` que contenha uma **função recursiva** que receba como parâmetros dois inteiros positivos k e n e retorna o resultado de k^n. Na sua implementação, você **deve** utilizar apenas multiplicações. Em seguida, inclua no seu `main.cpp` uma instrução para testar essa função. O programa deve ler do usuário os valores de k e n, chamar a função e imprimir na tela o valor retornado pela função.

Exemplo de execução do programa:

Digite k: **2**
Digite n: **3**
8

3. O máximo divisor comum (MDC) dos inteiros x e y é o maior divisor inteiro comum a x e y. Por exemplo, o MDC de 16 e 36 é o 4, enquanto que o MDC de 30 e 54 é o 6. Escreva uma classe `mdc` que contenha uma **função recursiva** que retorna o máximo divisor comum de x e y. Em seguida, inclua no seu `main.cpp` uma instrução para testar essa função. O programa deve ler do usuário os valores de x e y, chamar a função e imprimir na tela o valor retornado pela função.

Exemplo de execução do programa:

Digite x: **16**
Digite y: **36**
4

4. Crie e disponibilize um arquivo `makefile` para o compilação do seu programa.

Considerações!

- Para este exercício você pode criar um único projeto que inclua todas as classes solicitadas;
- Todos os exercícios devem conter `.h`, `.cpp`, e você pode criar apenas um único `main.cpp` para testar todas as funções implementadas;
- O seu `main.cpp` deve conter, minimamente, instruções para criação (instanciação de objetos) e chamadas das funções implementadas.