QUESTÃO 04 (bônus – 0,5 pontos) (Arthur Fernandes, 2025)

Com o código abaixo, complete as **linhas 10, 14, 23, 36, 45, 72 e 78** que faltam para que o programa funcione corretamente, realizando a divisão modular e o cálculo da congruência **H ÷ G (mod Zn)** seguido de **a^x mod n1**, aplicando o **Pequeno Teorema de Fermat** ou o **Teorema de Euler**, conforme o caso.

```c
#include <stdio.h>

#ifdef _WIN32
#include <windows.h>
#endif

// Função para calcular o máximo divisor comum (MDC) com exibição dos passos
int mdcComPassos(int a, int b) {
    int resto;
    while ([1] != 0) {
        resto = a % b;
        printf("Algorítmo de Euclides: %d mod %d = %d\n", a, b, resto);
        a = b;
        b = [2];
    }
    return a;
}
```

```c
int inversoModular(int a, int m) {
    int m0 = m, t, q;
    int x0 = 0, x1 = 1;
    int A = a, B = m;

    [3](a, m);

    while (m != 0) {
        q = a / m;
        t = m;
        m = a % m;
        a = t;

        t = x0;
        x0 = x1 - q * x0;
        x1 = t;
    }
    if (x1 < 0)
        [4] += m0;
    printf("\nSubstituindo, temos que o inverso de %d em %d é %d.\n\n", A, B, x1);
    return x1;
}
```

```c
int powMod(int base, int exp, int mod) {
    long long res = 1;
    long long b = base % mod;
    while (exp > 0) {
        if ([5])
            res = (res * b) % mod;
        b = (b * b) % mod;
        exp >>= 1;
    }
    return (int)res;
}
```

```c
int main() {
#ifdef _WIN32
    SetConsoleOutputCP(CP_UTF8);
#endif

    int H, G, Zn, x, n1;

    printf("Insira H: ");
    scanf("%d", &H);
    printf("Insira G: ");
    scanf("%d", &G);
    printf("Insira Zn: ");
    scanf("%d", &Zn);
    printf("Insira x: ");
    scanf("%d", &x);
    printf("Insira n1: ");
    scanf("%d", &n1);
    printf("\n");

    int inverso = [6](G, Zn);
    int a = (H * inverso) % Zn;

    printf("Fazendo a multiplicação modular: %d * %d mod %d ≡ %d\n", H, inverso, Zn, a);
    printf(" Sendo %d o inverso de %d.\n", inverso, G);

    int resultado = [7](a, x, n1);
    printf("Valor final da congruência: %d\n", resultado);

    return 0;
}
```

Com o código completo e preenchido corretamente, qual seria a saída com os valores: H: 7, G: 3, Zn: 11, x: 10, n1: 13

2. Considere o **código abaixo**, que realiza o cálculo da divisão modular H ÷ G (mod Zn) e depois computa a^x mod n1, aplicando o Pequeno Teorema de Fermat ou o Teorema de Euler, conforme a natureza de n1, classifique como Verdadeiro (V) ou Falso (F) cada uma das afirmativas a seguir:

```c
int inversoModular(int a, int m) {
    int m0 = m, t, q;
    int x0 = 0, x1 = 1;
    while (m != 0) {
        q = a / m;
        t = m;
        m = a % m;
        a = t;

        t = x0;
        x0 = x1 - q * x0;
        x1 = t;
    }
    if (x1 < 0)
        x1 += m0;
    return x1;
}
```

( ) O algoritmo de Euclides estendido é utilizado para calcular o **inverso modular** de um número.

( ) Se mdc(G, Zn) ≠ 1, o programa ainda consegue encontrar o inverso de G em Zn.

( ) A operação (H * inverso) % Zn representa a **divisão modular de H por G**.

( ) Se n1 for primo, o código aplica o **Pequeno Teorema de Fermat** para simplificar o cálculo de a^x mod n1.

( ) A função powMod implementa o cálculo de potência modular utilizando multiplicações diretas sem otimização.

( ) Quando o resultado do inverso é negativo, o código ajusta o valor somando o módulo m0.

( ) O cálculo de fi(n1) (função totiente de Euler) é utilizado apenas quando n1 **não é primo**.