1. [2 pontos]

```
/*
 * Arquivo: q1.c
 * -------------------
 * Este programa é uma solução da Questao 1 da primeira prova.
 */
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
main()
{
    int n, x, xx, f, ff;
    bool inconsistente = FALSE;
    printf("Qual 'e o numero de numeros a serem lidos? ");
    n = GetInteger();
    printf("Primeiro numero? ");
    xx = GetInteger();
    printf("Proximo numero? ");
    x = GetInteger();
    n -= 2;
    if (xx < x) f = 1;
    else if (x == xx) f = 0;
    else f = -1;
    xx = x;
    while (n > 0) {
        printf("Proximo numero? ");
        x = GetInteger();
        n--;
        if (xx < x) ff = 1;
        else if (x == xx) ff = 0;
        else ff = -1;
        if (f == 0) f = ff;
        else if (f * ff < 0) {
            inconsistente = TRUE;
            break;
        }
        xx = x;
    }
    while (n > 0) {
        printf("Proximo numero? ");
        x = GetInteger();
        n--;
    }
    if (inconsistente) printf("A sequencia nao 'e monotonica.\n");
    else if (f > 0) printf("A sequencia 'e crescente.\n");
    else if (f < 0) printf("A sequencia 'e decrescente.\n");
    else printf("A sequencia 'e constante.\n");
}
```

2. [2 pontos]

```
/*
 * Arquivo: q2.c
 * -------------------
 * Este programa é uma solução da Questao 2 da primeira prova.
 */
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
main()
{
    int n, x, xx, r;
    printf("Qual 'e o numero de numeros a serem lidos? ");
    n = GetInteger();
    printf("Primeiro numero? ");
    xx = GetInteger();
    n--;
    while (n > 0) {
        printf("Proximo numero? ");
        x = GetInteger();
        n--;
        while (x > 0)
          { r = xx % x; xx = x; x = r; }
    }
    printf("O maximo divisor comum 'e %d.\n", xx);
}
```

3. [2 pontos]

```
/*
 * Arquivo: q3.c
 * -------------------
 * Este programa é uma solução da Questao 3 da primeira prova.
 */
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
#include <math.h>
double cos_aprox(double x, int n);
/* Este é apenas um programa teste. */
main()
{
    int n;
    double x;
    printf("n = ");
    n = GetInteger();
    printf("x = ");
    x = GetReal();
    printf("cos %g (%d parcelas) = %g [math.h: cos %g = %g]\n",
           x, n, cos_aprox(x, n), x, cos(x));
}
/* Esta é a funcao que se pedia na questao. */
double cos_aprox(double x, int n)
{
    double soma = 1, termo = 1;
    int k;
    for (k = 1; k <= n; k++) {
        termo *= (-1) * x * x / (2 * k * (2 * k - 1));
        soma += termo;
    }
    return soma;
}
```

4. [2 pontos]

```
/*
 * Arquivo: q4.c
 * -------------------
 * Este programa é uma solução da Questao 4 da primeira prova.
 */
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
main()
{
    int k, m, n, bin = 1;
    printf("k = ");
    k = GetInteger();
    printf("m = ");
    m = GetInteger();
    printf("C(%d, %d) = %d\n", k, k, 1);
    for (n = k + 1; n <= m; n++) {
        bin = bin * n / (n - k);
        printf("C(%d, %d) = %d\n", n, k, bin);
    }
}
```

5. [2 pontos]
(*i*)

```
/*
 * Arquivo: q5.c
 * -------------------
 * Este programa é uma solução da Questao 5(i) da primeira prova.
 */
#include <stdio.h>
#include "genlib.h"
#include "simpio.h"
main()
{
    int m, n, a, aa, b, bb, c, d, q, r, t;
    printf("m = ");
    m = GetInteger();
    printf("n = ");
    n = GetInteger();
    aa = b = 1; a = bb = 0;
    c = m; d = n;
    while (TRUE) {
        q = c / d; r = c % d;
        if (r == 0) break;
        c = d; d = r;
        t = aa; aa = a; a = t - q * a;
        t = bb; bb = b; b = t - q * b;
    }
    printf("a = %d, b = %d, d = %d\n", a, b, d);
    printf("Temos (%d) * %d + (%d) * %d = %d\n", a, m, b, n, a * m + b * n);
}
```

(*ii*) Chamemos de (*) as condições válidas antes da execução do Passo 4. Para distinguir o valor de nossas variáveis antes e depois da execução das atribuições do Passo 4, vamos escrever a_1, a'_1, b_1, b'_1, etc, para os valores das variáveis a, a', b, b', etc, *depois* da execução do Passo 4. Temos, portanto,

$$c_1 = d \qquad d_1 = r,$$

e

$$a'_1 = a \qquad a_1 = a' - qa$$

e, finalmente,

$$b'_1 = b \qquad b_1 = b' - qb.$$

Verifiquemos agora as identidades dadas. Temos

$$a_1 m + b_1 n = (a' - qa)m + (b' - qb)n = a'm + b'n - q(am + bn) = c - qd = r = d_1.$$

Ainda,

$$a'_1 m + b'_1 n = am + bn = d = c_1.$$

Ademais, temos que $d_1 = r > 0$. Finalmente, verifiquemos que $\text{mdc}(c_1, d_1) = \text{mdc}(m, n)$. Temos $\text{mdc}(c_1, d_1) = \text{mdc}(d, r)$. Ainda, temos $c = qd + r$ e $r = c - qd$. Podemos ver que, portanto, o conjunto de divisores comuns a c e d são os mesmos que os divisores comuns a d e r. Daí segue que $\text{mdc}(c_1, d_1) = \text{mdc}(d, r) = \text{mdc}(c, d) = \text{mdc}(m, n)$. Note que última identidade é assumida em (*).
Assim, todas as condições exigidas são de fato verificadas após a execução do Passo 4.

(*iii*) As condições (*) são válidas ao iniciarmos a execução do algoritmo. Pelo item (*ii*), sabemos que a identidade $am + bn = d$ é preservada ao longo da execução do algoritmo. Finalmente, se $r = 0$ ocorre em algum ponto do algoritmo, então $\text{mdc}(c, d) = d$ neste ponto, pois d divide c. Entretanto, também sabemos que a condição $\text{mdc}(c, d) = \text{mdc}(m, n)$ é preservada ao longo da execução do algoritmo. Assim, ao ocorrer $r = 0$, temos que $\text{mdc}(m, n) = \text{mdc}(c, d) = d$, como queríamos demonstrar.