0.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MIN_FAIXA 20
#define MAX_FAIXA 80
#define N 20

int main ()
{
    int i, media, n_maiores, n_menores;
    int valores [N];

    srand (time (NULL));

    /* Gera N ints aleatórios e calcula a média. */
    media = 0;
    for (i = 0; i < N; i++)
    {
        /* Para gerar valores inteiros em um intervalo dado: */
        valores [i] = MIN_FAIXA + (rand () % (MAX_FAIXA-MIN_FAIXA+1));
        media += valores [i];
    }
    media /= N;
    printf ("Media: %d\n", media);

    /* Mostra cada valor. */
    n_maiores = 0;
    n_menores = 0;
    for (i = 0; i < N; i++)
    {
        printf ("%d\t", valores [i]);
        if (valores [i] > media)
        {
            n_maiores++;
            printf ("maior\n");
        }
        else if (valores [i] < media)
        {
            n_menores++;
            printf ("menor\n");
        }
        else
            printf ("igual\n");
    }

    printf ("Maiores: %d, menores: %d\n", n_maiores, n_menores);

    return (0);
}
```

1.

```
/* Neste exercício, é interessante observar como fazer para não guardar os
   valores negativos: você pode criar um vetor de tamanho N mas usar somente
   algumas das posições (aqui, n_positivos vai contar as posições usadas). */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 20
#define MIN_FAIXA -40
#define MAX_FAIXA 40

int main ()
{
    int i, n_positivos;
    float media;
    float val, valores [N];

    srand (time (NULL));

    /* Gera N valores e calcula a média dos positivos. */
    media = 0;
    n_positivos = 0;
    for (i = 0; i < N; i++)
    {
        /* Gerando valores float em um intervalo. */
        val = (rand () / (float) RAND_MAX) * (MAX_FAIXA-MIN_FAIXA) + MIN_FAIXA;
        if (val > 0)
        {
            valores [n_positivos] = val;
            media += valores [n_positivos];
            n_positivos++;
        }
    }
    media /= n_positivos;
    printf ("Media: %.2f\n", media);

    /* Mostra cada valor. */
    for (i = 0; i < n_positivos; i++)
    {
        printf ("%.2f\t", valores [i]);
        if (valores [i] > media)
            printf ("maior\n");
        else if (valores [i] < media)
            printf ("menor\n");
        else
            printf ("igual\n");
    }

    return (0);
}
```

2.

a)

```
#include <stdio.h>
#include <time.h>

#define N 10

int main ()
{
    int i;
    int valores [N];

    /* Lê N valores. */
    for (i = 0; i < N; i++)
        scanf ("%d", &(valores [i]));

    /* Mostra os valores na ordem inversa. */
    printf ("\n");
    for (i = N-1; i >= 0; i--)
        printf ("%d\n", valores [i]);

    return (0);
}
```

b)

```
#include <stdio.h>
#include <time.h>

#define MAX_N 1024

int main ()
{
    int i;
    int n;
    int valores [MAX_N];

    printf ("N:");
    scanf ("%d", &n);

    /* Lê N valores. */
    for (i = 0; i < n; i++)
        scanf ("%d", &(valores [i]));

    /* Mostra os valores na ordem inversa. */
    printf ("\n");
    for (i = n-1; i >= 0; i--)
        printf ("%d\n", valores [i]);

    return (0);
}
```

3.

a)

```
/* Usaremos um "truque", que envolve um vetor de 10 posições. Cada posição é um
   contador para o número de vezes que um dígito ocorreu no primeiro número.
   Depois, vamos subtraindo 1 da contagem para cada dígito do 2o número. No
   final, todos os contadores devem estar novamente em 0. */

int ehParDeFoolano (unsigned int n1, unsigned int n2)
{
    int i;
    int digitos [10];

    /* Inicia a contagem em 0. */
    for (i = 0; i < 10; i++)
        digitos [i] = 0;

    /* Conta cada dígito de n1. */
    while (n1 > 0)
    {
        digitos [n1 % 10]++;
        n1 /= 10;
    }

    /* Agora verifica n2. */
    while (n2 > 0)
    {
        int d = n2 % 10;
        digitos [d]--;
        n2 /= 10;
    }

    /* Checa se tem algum contador com valor diferente de 0. */
    for (i = 0; i < 10; i++)
        if (digitos [i] != 0)
            return (0);

    return (1);
}
```

b)

```
/* Apesar de permitir um retorno antes de verificar todos os dígitos, a solução
   sem vetores precisa decompor ambos os números entre 1 e 10 vezes, enquanto a
   solução com os vetores permite sempre decompor os números uma única vez.
   A solução com vetores usa mais memória, mas é mais eficiente em tempo. */

int contaOcorrencias (unsigned int n, unsigned int digito)
{
    int count = 0;

    /* "Destrói" o número enquanto conta as ocorrências do dígito. */
    while (n)
    {
        if (n%10 == digito)
            count++;
        n /= 10;
    }

    return (count);
}

int ehParDeFoolano (unsigned int n1, unsigned int n2)
{
    int digito;

    /* Conta as ocorrências de cada dígito. */
    for (digito = 0; digito <= 9; digito++)
        if (contaOcorrencias (n1, digito) != contaOcorrencias (n2, digito))
            return (0); /* Pode parar imediatamente! */

    return (1);
}
```

4.

```
#include <stdio.h>

/* Como ainda não vimos alocação dinâmica, vamos criar um vetor com o maior
   valor permitido para N, e usar somente parte dele. */
#define MAX_N 100

int main ()
{
    int tam_secoes [MAX_N];
    int i, n, tam_total, soma_esq;

    scanf ("%d", &n);

    // Lê os tamanhos das seções.
    // Ao mesmo tempo, já calcula a soma de todas as seções.
    tam_total = 0;
    for (i = 0; i < n; i++)
    {
        scanf ("%d", &tam_secoes [i]);
        tam_total += tam_secoes [i];
    }

    // Agora, temos uma meta: cada país terá exatamente a metade da soma total.
    tam_total /= 2; // Só precisamos ver até a metade!
    soma_esq = 0;

    // Note que aqui nem precisaria do i < n, mas é bom manter por segurança.
    for (i = 0; soma_esq < tam_total && i < n; i++)
        soma_esq += tam_secoes [i];

    /* Quando chegar aqui, o valor de i vai ser 1 a mais que a última
       posição somada (por causa do i++). Não precisamos ajustar porque a
       especificação tem uma "pegadinha": as posições lá começam em 1! */
    printf ("%d\n", i);

    return (0);
}
```

5.

```
#include <stdio.h>

/* Como ainda não vimos alocação dinâmica, vamos criar um vetor com o maior
   valor permitido para N, e usar somente parte dele. */
#define MAX_N 100

int main ()
{
    int tabuleiro [MAX_N];
    int i, n;

    scanf ("%d", &n);

    // Lê o tabuleiro.
    for (i = 0; i < n; i++)
        scanf ("%d", &tabuleiro [i]);

    // Repassa o tabuleiro e mostra a soma dos valores em 3 posições.
    // Por simplicidade, tratamos a primeira e a última posição fora do loop.
    printf ("%d ", tabuleiro [0] + tabuleiro [1]); // n precisa ser >= 2.
    for (i = 1; i < n-1; i++)
        printf ("%d ", tabuleiro [i-1] + tabuleiro [i] + tabuleiro [i+1]);
    printf ("%d ", tabuleiro [n-2] + tabuleiro [n-1]); // Final.

    return (0);
}
```