0.

a)
Se a única tarefa é imprimir os valores, não é preciso um vetor. Podemos fazer da seguinte forma:

```
repete N vezes:
      lê um número
      se o número estiver nos limites da faixa
            mostra o número
```

b)
Se só queremos saber a soma total, não precisamos de um vetor. Podemos usar a velha técnica do acumulador: iniciamos uma variável em 0 e somamos a ela cada um dos valores da sequência.

c)
Ainda não conhecemos um algoritmo para ordenar valores, mas já fizemos exercícios que envolviam colocar 3 valores lidos em ordem. Pense agora em um algoritmo para 4 valores. Note que se formos lendo os valores um a um, no momento em que um valor é lido, não sabemos se ele é menor ou maior que os valores anteriores ou os próximos, ou seja, nós precisamos conhecer todos os valores antes de escrevê-los em ordem. Como fazer isso com N variáveis seria inviável, a solução deve envolver vetores, de alguma forma.

d)
Não precisamos de um vetor. Podemos armazenar em uma variável o menor elemento, e ir atualizando esta variável conforme os valores são lidos. Algo assim:

```
menor = o maior número possível, ou o primeiro elemento de entrada
para cada elemento de entrada
      lê um número
      se o número é menor que o menor
            menor = número
no fim, a variável menor contém o menor número
```

e)
Não precisamos de um vetor. Podemos armazenar os dois menores valores, e ir atualizando estas variáveis conforme os valores são lidos. Algo assim:

```
menor1 = o maior número possível
menor2 = o maior número possível
para cada elemento de entrada
      lê um número
      se o número é menor que menor1
            menor2 = menor1
            menor1 = número
      senão, se o número é menor que menor2
            menor2 = número
no fim, menor1 é o menor número, menor2 o segundo menor
```

Repare que isso funciona bem para o segundo menor elemento, mas não para o n-ésimo menor elemento. Se queremos o n-ésimo menor elemento, precisamos lembrar de todos os n menores elementos da sequência (ou dos tamanho-n maiores), e quanto maior for o n, mais desajeitada será a solução. Neste caso, precisaríamos de um vetor.

1.

```
#include <stdio.h>

int main ()
{
    int v [10];
    int i;

    for (i = 0; i < 10; i++)
        v [i] = i;

    for (i = 0; i < 10; i++)
        printf ("%d ", v [i]);

    return (0);
}
```

2.

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

#define N 10

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

    srand (time (NULL));

    for (i = 0; i < N; i++)
    {
        vetor [i] = rand () % N;
        printf ("%d\n", vetor [i]);
    }

    printf ("-----\n");

    for (i = 0; i < N; i++)
        if (vetor [i] == i)
            printf ("%d\n", i);

    return (0);
}
```

3.

```
#include <stdio.h>

#define N 10

int main ()
{
    int n, i;
    int numeros [N];

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

    /* Imprime os que estão em posições pares. */
    for (i = 0; i < N; i+=2)
        printf ("%d\n", numeros [i]);

    /* Imprime os que estão em posições ímpares. */
    for (i = 1; i < N; i+=2)
        printf ("%d\n", numeros [i]);

    return (0);
}
```