1 Sequential search

(1.1) You are given a collection of data items, stored in an array. Write a routine

```
int find (int x, int n, int array a[] )
```

which returns some index `i` such that `x == a[i]`, if it exists, otherwise returns `-1` when `x` is not found.

That is, we want the *location* of `x` in the array.

(1.2) The foolproof method is *sequential search.*

```
int find (int x, int n, int array a[] )
{
  int i;
  for (i=0; i<n; ++i)
    if ( x == a[i] )
      return i;

  return -1;
}
```

(1.3) *Efficiency.* The study of algorithms is usually concerned with efficiency, usually meaning *cost* or *runtime* as measured against the size of the data. In this case the data is `a[0..n-1]` and its size is n integers.

The runtime is, of course, the time in microseconds taken by this routine.

In the worst case `x` is not found and all the data is scanned, with `n` iterations.

(1.4) Definition *Let $f, g : \mathbb{N} \to [0, \infty)$ be two functions. Assume that there is an index N such that for all $k \geq N$, $g(k) > 0$.*

$$f \quad \textit{is} \quad O(g)$$

if

$$\overline{\lim_{n\to\infty, n\geq N}} \frac{f(n)}{g(n)} < \infty$$

This definition is nonstandard and not useful. The proper formulation is: There exists a nonnegative constant c such that for all $n \geq N$,

$$f(n) \leq cg(n).$$

(1.5) The runtime of sequential search (1.2) is bounded by $a+bk$ where a and b are estimates based on the computing power applied and k is the number of iterations. Since $k \leq n$, the runtime is $O(n)$.