The Algorithm for Finding the \(k\)th Smallest in Expected Linear Time (Non-Recursive)

Let \(T\) be the type of the elements in array \(a\). As the algorithm runs, at all times the \(k\)th smallest element of \(a\) lies between positions \(left\) and \(right\), inclusive. The algorithm terminates when \(left == right\), at which time both must equal \(k\).

```
T select( T[] a, Integer k)
    left = 1;
    right = a.length;
    while ( left < right )
        q = partition( a, left, right);
        if ( k < q )
            right = q - 1;
        else if ( k > q )
            left = q + 1;
        else
            left = right = q;
    return a[q];
```
A Recursive Implementation of The Algorithm for Finding the $k$th Smallest in Expected Linear Time

The recursive version of $select()$ below is initiated by invoking $select( a, k, 1, a.length)$. It assumes the array $a$ is passed by reference and the integer arguments by value, as in C, C++, or Java.

When $select(a, k, left, right)$ is called, the $k$th smallest element of the entire array $a$ must lie between positions $left$ and $right$, inclusive. $select()$ completes the computation of the $k$th smallest element and returns it.

```
T select( T[] a, Integer k, Integer left, Integer right) {
    if ( left < right )
        q = partition( a, left, right);
        if ( k < q )
            return select( a, k, left, q-1);
        else if ( k > q )
            return select( a, k, q+1, right);
        else
            return a[q];
    else
        return a[left];
}
```