`arrayutil.c0` - utility functions for integer arrays
15-122, Principles of Imperative Computation

```
/* is_in: x in A[lo..hi) */
bool is_in(int x, int[] A, int lo, int hi)
/*@requires 0 <= lo  && lo <= hi && hi <= \length(A); @*/ ;

/* is_sorted: A[lo..hi) SORTED */
bool is_sorted(int[] A, int lo, int hi)
/*@requires 0 <= lo  && lo <= hi && hi <= \length(A); @*/ ;

/* swap(A, i, j) has the effect of switching A[i] and A[j] */
void swap(int[] A, int i, int j)
/*@ requires 0 <= i && i < \length(A) && 0 <= j && j < \length(A); @*/ ;
```

Comparing array segments to a value

x > A[lo..hi) means that, for all i ∈ [lo, hi), we have that x > A[i]

```
/* gt_seg: x > A[lo..hi) */
bool gt_seg(int x, int[] A, int lo, int hi)
/*@requires 0 <= lo && lo <= hi && hi <= \length(A); @*/ ;

/* ge_seg: x >= A[lo..hi) */
bool ge_seg(int x, int[] A, int lo, int hi)
/*@requires 0 <= lo && lo <= hi && hi <= \length(A); @*/ ;

/* lt_seg: x < A[lo,hi) */
bool lt_seg(int x, int[] A, int lo, int hi)
/*@requires 0 <= lo && lo <= hi && hi <= \length(A); @*/ ;

/* le_seg: x <= A[lo..hi) */
bool le_seg(int x, int[] A, int lo, int hi)
/*@requires 0 <= lo && lo <= hi && hi <= \length(A); @*/ ;
```

Comparing two array segments

A[lo1..hi2) > B[lo2..hi2) means that,
for all i ∈ [lo1, hi1) and for all j ∈ [lo2, hi2), we have that A[i] > B[j]

```
/* gt_segs: A[lo1..hi1) > B[lo2..hi2) */
bool gt_segs(int[] A, int lo1, int hi1, int[] B, int lo2, int hi2)
/*@requires 0 <= lo1 && lo1 <= hi1 && hi1 <= \length(A); @*/
/*@requires 0 <= lo2 && lo2 <= hi2 && hi2 <= \length(B); @*/ ;

/* ge_segs: A[lo1..hi1) >= B[lo2..hi2) */
bool ge_segs(int[] A, int lo1, int hi1, int[] B, int lo2, int hi2)
/*@requires 0 <= lo1 && lo1 <= hi1 && hi1 <= \length(A); @*/
/*@requires 0 <= lo2 && lo2 <= hi2 && hi2 <= \length(B); @*/ ;

/* lt_segs: A[lo1..hi1) < B[lo2..hi2) */
bool lt_segs(int[] A, int lo1, int hi1, int[] B, int lo2, int hi2)
/*@requires 0 <= lo1 && lo1 <= hi1 && hi1 <= \length(A); @*/
/*@requires 0 <= lo2 && lo2 <= hi2 && hi2 <= \length(B); @*/ ;

/* le_segs: A[lo1..hi1) <= B[lo2..hi2) */
bool le_segs(int[] A, int lo1, int hi1, int[] B, int lo2, int hi2)
/*@requires 0 <= lo1 && lo1 <= hi1 && hi1 <= \length(A); @*/
/*@requires 0 <= lo2 && lo2 <= hi2 && hi2 <= \length(B); @*/ ;
```