**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); @\*/ ;**