Question 2. Emergency Ward Triage (65 marks)

In this assignment you will complete the implementation of a system to triage patients in a hospital emergency ward. Patients are triaged according to both medical priority and wait time. Priorities are positive integers; the highest priority is 1. Normally patients are seen in priority order. However, if there are patients who have waited longer than a specified time (maxWait), they are seen first, in order of their arrival.

The main operations you must support are to add a patient to the system when they arrive, and to remove the next patient to be seen from the system when a physician becomes available. Both operations must be O(log n), where n is the number of patients in the system.

To achieve this, the system will use two location-aware priority queues implemented with min heaps: one based on medical priority and the other on arrival time. The heaps use ArrayLists to store the underlying binary trees.

The key problem here is that when a patient is removed from one queue they must also be removed from the other. Location-aware priority queues allow us to do this efficiently.

The Java package implementing this system consists of 12 classes, of which 10 are fully implemented for you. Your task is to complete the implementation of two classes:

Class APQ: A location-aware adaptable priority queue based upon a min-heap.
Methods you must implement: offer, remove, poll, downheap, upheap, swap

Class PatientTriage: Uses APQ to triage patients.
Methods you must implement: remove

Note: Both of your priority queues are based upon heaps, which in turn are implemented as array lists, which in turn are implemented as arrays. We are using array lists only to take advantage of the automatic resizing capability. APQ.offer should use the add(e) method of the ArrayList class to initially append elements to the array, and APQ.remove should use the remove(p) method of ArrayList to remove the last element of the array, where p is a position. However, we are NOT using the add(i, e) or remove(i) methods to automatically shift later elements to the right or left when adding or removing elements in the middle of the list. These operations will NOT preserve the heap property of your heap. Instead you must use the procedures discussed in class for adding or removing an element from a heap.

The package includes a test program **testPatientTriage** that provides a test case. You should, however, test your program using a broader range of test cases. Pay particular attention to boundary conditions.