cpts350 hw2'

Re-organizing elements in an array forms a large class of tricky problems in computer science. Some of these problems seek linear time algorithms to solve. In particular, a one-pass and in-place is the trickest to obtain:

- *One-pass.* You create a constant (like 3) number of pointers (indexed to the array) and move each pointer in only one direction through out the algorithm (i.e., the first pointer always moves to the left and the second pointer always moves to the right, etc.); of course the less of the number the better.

- *In-place.* Your algorithm need only constant amount of auxillary memory in addition to the given array.

You may now carefully run the Partition algorithm in quicksort and confirm that the Partition is indeed one-pass and in-place.

In almost all such re-organizing problems, each element in the array is assumed to take one unit space and so is an integer (as well as a pointer). Many students are too smart in solving such a problem by putting a marker on an array element in the array – this will spend non-constant extra memory and you dont have it! To re-organize an array, it is neccessary to move the elements around in the array and `swap`(i, j) which swaps $A[i]$ with $A[j]$ in array A, is more than enough to serve the purpose (you dont need any other fancy element moving operations).

Let A be an array of n babies and hence all elements are distinct.
1. Each baby has brown hair or purple hair. Design a one-pass and in-place and linear time algorithm to sort the babies according to their hair color: brown hair babies followed by purple hair babies. In your algorithm, you must use only two pointers. (a). Please tell me explicitly what the constraints are we put on those two pointers (i.e., you interpret intuitively, and possibly with some drawings on the meanings of "one-pass", "in-place", and "linear-time".). (b). The thought process of designing an algorithm is really important — you have to walk in the right direction at the first trial. When you design the algorithm, you have to bear in mind simontaneously the three constraints ("one-pass", "in-place", and "linear-time") while moving the two pointers in your mind on the array of babies. Please write down the thought process of your design of the algorithm and in particular, mentioning among the three constraints, which one you consider the first and why. (c). Please write on almost working psuedo-code with comment.

2. (hard) Each baby has brown hair or purple hair or black hair. Design a one-pass and in-place and and linear time algorithm to sort the babies according to their hair color: brown hair babies followed by purple hair babies and then followed by black hair babies. Can you use only three pointers? If you need more than three pointers, how many do you need? (Remember that each pointer is one-pass.) I need almost working psuedo-code.