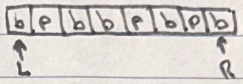
Hw 4 pg.1

1) Let A be an array of n babies and hence all elements are distinct each baby has brown or purple hair. Design a one-pass, in place, linear time alg to sort. Brown hair followed by purple. Only use 2 ptr's.
   a) Tell what constraints are we put on these 2 ptrs
   b) Write down thought process.
   c) Write in pseudo-code.



c alg.

```
def sortbabies (babies)
    leftptr = 0        # left ptr starts front of array
    rightptr = length(babies) -1   # right ptr starts back
      While (leftptr < rightptr)   # until ptrs cross (limit to 1 pass)
          while ( babies[leftptr] == brown ) #move left until purple found
                and leftptr < len(babies)  or until end array
              leftptr ++
          while ( babies[rightptr] == purple) #do same but opposite
                and rightptr ≥ leftptr)          for right
              rightptr ++
          if ( leftptr < rightptr )  #as long as ptrs haven't crossed we
                                                          swap
              Swap ( babies[leftptr], babies[rightptr])
      return babies
```

a) The constraints we put on the two ptrs are that they can't cross eachother (ruin one-pass constraint & linear time constraint) and can't go outside of the array bounds (create new memory). Additionally they can't be used to write new memory (in-place constraint)

b) My thought process started by figuring out how to only do one pass on the array. As long as I did this keeping linear time and not using new memory would be easy. I figured I could loop through and ~~stop~~ my ptrs when a swap could happen as long as my ptrs checked position (not crossed) and didn't leave array. Thus I made a nested loop that did one pass and checked the given criteria every loop.