PROGRAMMING ASSIGNMENT #3
CS 2223 D-TERM 2020
PALINDROMES, INVERSIONS, AND BINARY REFLECTED GRAY CODE

SEVENTY-FIVE POINTS
DUE: MONDAY, APRIL 20, 2020 2PM (EXTENDED)

1. (20 Points) A ***palindrome*** is a word, phrase, or sequence that reads the same backward as forward. Write a recursive Java program/procedure which determines whether an input sequence (string) from the keyboard is a palindrome or not. (You can assume the input will be shorter than 256 characters.)
   Call your program/project 'palindromecheck'.

   (3 Bonus Points) Make your program case insensitive, and have your program ignore white space and punctuation:

"Never odd or even" "Able was I ere I saw Elba" "A man, a plan, a canal: Panama!"

2. (25 Points) Let $A[0..n-1]$ be an array of real numbers (or any ordered set). A pair $(A[i], A[j])$ is said to be an ***inversion*** if these numbers (elements) are out of order, i.e., $i < j$ but $A[i] > A[j]$. Note that this pair need not be adjacent. The array/sequence (3, 2, 1) contains *three* inversions: (3,2), (2,1), and (3,1).

   (10 Points) Write a program with a naïve $O(n^2)$ algorithm that counts the number of inversions in such an array A. Hint: BUBBLESORT is $O(n^2)$.
   Hint^2: What does BUBBLESORT do with simple inversions?

   Call your program/project 'easyinversioncount'.

   (15 Points) Write a program with a $O(n \log n)$ algorithm that counts the number of inversions in such an array A. Hint: MERGESORT is $O(n \log n)$, and this is problem #9 in section 5.1 of Levitin which covers MERGESORT. You do have to notice something in your counting... a bit more than in BUBBLESORT.

   Call your program/project 'fastinversioncount'.

3. (30 Points) Binary Reflected Gray Code

Professor Wolfe has four children: Alice, Bob, Chris, and Dylan. Each year for Arbor Day the family goes on a tree-planting picnic, and Professor Wolfe takes photos of the kids in every possible subset arrangement with the new trees they have planted as a backdrop. It's a lovely tradition. However, last year with newborn Dylan things got more complicated. In transitioning from photograph 7 (0111) to photograph 8 (1000), Professor Wolfe placed baby Dylan on the picnic blanket while all the other children got up and left the scene. Mayhem ensued. Bob and Chris started chasing each other with rakes, and Alice complained about having to move every time a photo was to be taken.

This year is going to be different. Instead of using a Base Two counting sequence to ensure that all the subset arrangements are made, Professor Wolfe is going to use the Binary Reflected Gray Code of order four. In this way, each transition from one subset of children to the next can be made by having just one child enter or leave the tableau:

Index	Gray Code	Child(ren) in Photo	Action
1	0001	Alice	Alice In
2	0011	Bob & Alice	Bob In
3	0010	Bob	Alice Out
4	0110	Chris & Bob	Chris In
5	0111	Chris & Bob & Alice	Alice In
6	0101	Chris & Alice	Bob Out
⋮	⋮	Etc.	Etc.

- (10 Points) Using Java, implement algorithm BRGC(n) on page 148 of Levitin to produce the Binary Reflected Gray Code of order n.

- (10 Points) Add to your routine (or write a separate one using the idea in Problem 9b on page 149 of Levitin) to create a sequence of names representing which child must move when.
  The partial sequence up to six is: Alice, Bob, Alice, Chris, Alice, Bob, ...

- (10 Points) Put it all together to complete the table above for all 15 photos Professor Wolfe wants to take. You can even start at 0=0000, if you'd like just the landscape with the new trees!

Call your program/project 'graycodesarefun'.
(Where have we seen this before? How many interesting patterns can *you* find?)

(5 Bonus Points) Can you extend your code to more children... Eve? Felix? Gerry? Helen? Ian? Jane? Karl...?