CSCI-398 Homework 1

In all problems you may write other functions that help to solve the problem.

Problem 1. Given three strings s, s1 and s2. You need to write a function that finds substrings s1 inside of the string s and replaces each substring s1 with substring s2.

Requirements:

1. Your function must do replacement "in place", i.e. inside string s itself. This means that you cannot use any additional space: you can use some local variables, strings of length s1 or s2, but not of length s. In short, space requirement is O(1) (considering that the length of s is much longer than length of s1 or s2). In case, when s2 is longer than s1, first figure out how long the final string must be, resize s to the new size, and then make replacements in place.
2. If s1 does not occur in s, return false.
3. If s1 overlaps itself, then return false and do not make any replacements. For example, if s = "ABRACABABARA", and s1 = "ABA", then s1 overlaps itself in the substring of s "ABABA" (the end of the first occurrence of ABA is the beginning of the second occurrence of ABA).
4. Time requirement is O(s.length*s1.length) and space requirement is O(1).
5. The declaration of this function is: *bool **replace(*****string &s, const string &s1, const string &s2*****)***.

Problem 2. Given a string *s* consisting of English alphabet lower case characters. Write a function that returns true if *s* is a rotation of a palindrome.

Definition: A rotation (or a circular shift) of a given string *s* = $x_1x_2...x_n$ is a string *t* = $x_ix_{i+1}...x_nx_1...x_{i-1}$, i.e. string *t* is a concatenation of a suffix of *s* and the remaining prefix of *s.*

Example: *t* = CDEAB is a rotation of string *s* = ABCDE.

Examples: **YAKKA** is a rotation of the palindrome KAYAK.

Requirements:
Let *n* be the length of a given string *s.*

1. Time complexity is O(n^2).
2. Space complexity is O(n).
3. The declaration of this function is: bool **palindromeRotation(const string &s)**

Problem 3. Given an array of integers (not sorted). The integers in the array are elements of an arithmetic sequence of size n. Write a function that takes such an integer array and sorts array in increasing order in place (do not use any additional data structures, and do not copy array) in linear time.
Requirements:

1. Space complexity is O(1).
2. Time complexity is O(n).
3. The header of this function must be: *void **sortInPlace(int *ptr, const int size)***

Definition: An **arithmetic sequence** is a sequence of numbers such that the difference between the consecutive terms is constant. It can be defined recursively:

The first element A_0 is given, and the i-th element of the sequence is calculated from the previous (i-1)th element using this formula:
$A_i = A_{i-1} + d$, where difference d is a constant.

Example: 7, 17, 27, 37, 47, 57, 67, 77 is an arithmetic sequence of size 8 with the first element 7 and difference 10 between two consecutive elements of the sequence.

Input example: 27 7 37 17 57 47 77 67
Sorted array: 7 17 27 37 47 57 67 77