Practice Problems for PE03

For **PE03**, you'll complete two short functions with **strings and loops**. These are similar to the Codingbat or PracticeIt exercises you may have completed in Java or Python. Below are some sample problems that you can use for practice.

1 THE WORD ENDS PROBLEM

Given a string and a non-empty word string, set result to a string made of each char just before and just after every appearance of the word in the string. Ignore cases where there is no char before or after the word, and a char may be included twice if it is between two words. Do not do any printing.

- for input of `"abcXY123XYijk", "XY"` → `"c13i"`
- for input of `"XY123XY", "XY"` → `"13"`
- for input of `"XY1XY", "XY"` → `"11"`

2 THE PLUS OUT PROBLEM

Given a string and a non-empty word string, set result to a version of the original string where all chars have been replaced by pluses ("+"), except for appearances of the word string which are preserved unchanged.

- for input of `"12xy34", "xy"` → `"++xy++"`
- for input of `"12xy34", "1"` → `"1+++++"`
- for input of `"12xy34xyabcxy", "xy"` → `"++xy++xy+++xy"`

3 THE STAR OUT PROBLEM

Set result to a version of the given string, where for every star (*) in the input string the star and the chars immediately to its left and right are gone. So "ab*cd" yields "ad" and "ab**cd" also yields "ad".

- for input of `"ab*cd"` → `"ad"`
- for input of `"ab**cd"` → `"ad"`
- for input of `"sm*eilly"` → `"silly"`

4 THE NIK NAK PROBLEM

Look for patterns like "nik" and "nak" in the input string–in other words a substring, length-3, starting with 'n' and ending with 'k'. Set result to a string where for all such words, the middle letter is gone, so "nikXnak" yields "nkXnk".

- for input of `"nikXnak"` → `"nkXzp"`
- for input of `"noknok"` → `"nknk"`
- for input of `"nnnoknok"` → `"nnnknk"`

5 THE SANDWICH PROBLEM

A sandwich is two pieces of bread with something in between. Set result to the string that is between the first and last appearance of "bread" in the given string, or set it to the empty string "" if there are not two pieces of bread.

- for input of `"breadjambread"` → `"jam"`
- for input of `"xxbreadjambreadyy"` → `"jam"`
- for input of `"xxbreadyy"` → `""`

6 THE REPEAT SEPARATOR PROBLEM

Given as input two strings, word and a separator, set result to a big string made of count occurrences of the word, separated by the separator string.

- for input of `"Word", "X", 3` → `"WordXWordXWord"`
- for input of `"This", "And", 2` → `"ThisAndThis"`
- for input of `"This", "And", 1` → `"This"`

7 THE REPEAT FRONT PROBLEM

Given as input a string and an int n, set result to a string made of the first n characters of the string, followed by the first n-1 characters of the string, and so on. You may assume that n is between 0 and the length of the string, inclusive (i.e. n >= 0 and n <= str.length()).

- for input of `"Chocolate", 4` → `"ChocChoChC"`
- for input of `"Chocolate", 3` → `"ChoChC"`
- for input of `"Ice Cream", 2` → `"IcI"`

8 THE REPEAT END PROBLEM

Given as input string and an int N, set result a string made of N repetitions of the last N characters of the string. You may assume that N is between 0 and the length of the string, inclusive.

- for input of `"Hello", 3 → "llollollo"`
- for input of `"Hello", 2 → "lolo"`
- for input of `"Hello", 1 → "o"`

9 THE MIX STRING PROBLEM

Given two input strings, A and B, create a bigger string made of the first char of A, the first char of B, the second char of A, the second char of B, and so on. Any leftover chars go at the end of the result.

- for input of `"abc", "xyz" → "axbycz"`
- for input of `"Hi", "There" → "HTihere"`
- for input of `"xxxx", "There" → "xTxhxexre"`

10 THE COUNT CODE PROBLEM

Set result to the number of times that the string "code" appears anywhere in the input given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.

- for input of `"aaacodebbb" → 1`
- for input of `"codexxcode" → 2`
- for input of `"cozexxcope" → 2`

11 THE FRONT TIMES PROBLEM

Given an input string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Set result to n copies of the front;

- for input of `"Chocolate", 2 → "ChoCho"`
- for input of `"Chocolate", 3 → "ChoChoCho"`
- for input of `"Abc", 3 → "AbcAbcAbc"`

12 The String Yak Problem

Suppose the string "yak" is unlucky. Given a string, set result to a version where all the "yak" are removed, but the "a" can be any char. The "yak" strings will not overlap.

- for input of `"yakpak"` → `"pak"`
- for input of `"pakyak"` → `"pak"`
- for input of `"yak123ya"` → `"123ya"`

13 The String Bits Problem

Given a string, set result to a new string made of every other char starting with the first, so "Hello" yields "Hlo".

- for input of `"Hello"` → `"Hlo"`
- for input of `"Hi"` → `"H"`
- for input of `"Heeololeo"` → `"Hello"`

14 The String Splosion Problem

Given a non-empty string like "Code" set result to a string like "CCoCodCode".

- for input of `"Code"` → `"CCoCodCode"`
- for input of `"abc"` → `"aababc"`
- for input of `"ab"` → `"aab"`

15 The String X Problem

Given a string, set result to a version where all the "x" have been removed. Except an "x" at the very start or end should not be removed.

- for input of `"xxHxix"` → `"xHix"`
- for input of `"abxxxcd"` → `"abcd"`
- for input of `"xabxxxcdx"` → `"xabcdx"`

16 The String Match Problem

Given 2 strings, a and b, set result to the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings.

- for input of `"xxcaazz", "xxbaaz" → 3`
- for input of `"abc", "abc" → 2`
- for input of `"abc", "axc" → 0`

17 The Last Two Problem

Given a string, set result to the count of the number of times that a substring length 2 appears in the string and also as the last 2 chars of the string, so "hixxxhi" yields 1 (we won't count the end substring).

- for input of `"hixxhi" → 1`
- for input of `"xaxxaxaxx" → 1`
- for input of `"axxxaaxx" → 2`

18 The Count YZ Problem

Given an input string, count the number of words ending in 'y' or 'z' –so the 'y' in "heavy" and the 'z' in "fez" count, but not the 'y' in "yellow". Make sure that your comparison is not case sensitive. We'll say that a y or z is at the end of a word if there is not an alphabetic letter immediately following it.

- for input of `"fez day" → 2`
- for input of `"day fez" → 2`
- for input of `"day fyyyz" → 2`

19 The Without String Problem

Given two input strings, base and remove, set result to a version of the base string where all instances of the remove string have been removed. This is not case sensitive. You may assume that the remove string is length 1 or more. Remove only non-overlapping instances, so with "xxx" removing "xx" leaves "x".

- for input of `"Hello there", "llo" → "He there"`
- for input of `"Hello there", "e" → "Hllo thr"`
- for input of `"Hello there", "x" → "Hello there"`

20 The Count Triple Problem

We'll say that a "triple" in a string is a char appearing three times in a row. Set result to the number of triples in the given string. The triples may overlap.

- for input of `abcXXXabc" → 1`
- for input of `"xxxabyyyycd" → 3`
- for input of `"a" → 0`

21 The Sum Digits Problem

Given a string, set result to the sum of the digits 0-9 that appear in the string, ignoring all other characters. Return 0 if there are no digits in the string.

- for input of `"aa1bc2d3" → 6`
- for input of `"aa11b33" → 8`
- for input of `"Chocolate" → 0`

22 The Same Ends Problem

Given a string, set result to the longest substring that appears at both the beginning and end of the string without overlapping. For example, given the input "abXab", then result "ab".

- for input of `"abXYab" → "ab"`
- for input of `"xx" → "x"`
- for input of `xxx" → "x"`

23 The Mirror Ends Problem

Given a string, look for a mirror image (backwards) string at both the beginning and end of the given string. In other words, zero or more characters at the very beginning of the given string, and at the very end of the string in reverse order (possibly overlapping). For example, the string "abXYZba" has the mirror end "ab".

- for input of `"abXYZba" → "ab"`
- for input of `"abca" → "a"`
- for input of `"aba" → "aba"`

24 The Max Block Problem

Given an input string, set result to the length of the largest "block" in the string. A block is a run of adjacent chars that are the same.

- for input of `"hoopla" → 2`
- for input of `"abbCCCddBBBxx" → 3`
- for input of `"" → 0`

25 The Not Replace Problem

Given an input string, set result to a string where every appearance of the lowercase word "is" has been replaced with "is not". The word "is" should not be immediately preceded or followed by a letter -- so for example the "is" in "this" does not count.

- for input of `"is test" → "is not test"`
- for input of `"is-is" → "is not-is not"`
- for input of `"This is right" → "This is not right"`