Start



is\_vowel -> character to bool, is\_vowel returns true when its argument is a vowel and false otherwise.



is\_consonant -> character to bool, is\_consonant returns true when its argument is a consonant and false otherwise.



ends\_with -> two strings, candidate and suffix (in that order) to bool

If candidate is an empty string and suffix is an empty string, ends\_with returns true. If candidate is an empty string and suffix is not an empty string, ends\_with returns false. If candidate ends with suffix, ends\_with returns true. Otherwise, it returns false.



ends\_with\_double\_consonant a string to bool

ends\_with\_double\_consonant returns true if the last two characters of the argument are both consonants, and equal to one another. It returns false otherwise.



ends\_with\_cvc string to bool,

ends\_with\_cvc returns true if the last three characters of the argument are a consonant, a vowel and then a consonant (hence the name, cvc). It returns false otherwise.



count\_consonants\_at\_front -> string to int

count\_consonants\_at\_front returns the number of consecutive constants at the beginning of the argument. For example, count\_consonants\_at\_front (std::string{"threw"}) is 3.



count\_vowels\_at\_back string to int

count\_vowels\_at\_back returns the number of consecutive vowels at the end of the argument. For example, count\_vowels\_at\_back(std::string{"free"}) is 2. The string passed as an argument may be empty.



contains\_vowel -> string to bool

contains\_vowel returns true if there is a vowel anywhere in the argument. It returns false otherwise. The string passed as an argument may be empty.



new\_ending -> string, a number and a string named candidate, suffix length and replacement, respectively (and in that order) to string

new\_ending creates a new string from candidate by removing its last suffix length characters and replacing them with replacement. new\_ending returns that new string. For example, new\_ending(std::string{"testing"}, 3, std::string{"ed"}) is "tested". You may assume that suffix length is always less than the length of candidate.