Week 7: In-Class Exercises

Please download the resource from *http://blue.smu.edu.sg/cs101/ice7-resource.zip*

1. **[Difficulty: *]** Implement the `average` function. The function takes in an array of numbers `values` (type: `int[]`) as its parameter. It will return the average of all the numbers.

2. **[Difficulty: *]** Implement a function called `count_odd_numbers()`. The function takes in 1 parameter, `numbers` (type: `int[]`). The function returns the number of integers whose value is odd.

3. **[Difficulty: *]** Implement a function called `get_median()`. The function takes an array of integers, and returns the median. You should re-use the `bubble_sort` function implemented in Q3.

4. **[Difficulty: **]** Implement a function called `exchange_pairs()`. The function takes a string, and switches the values pairwise: swap the value at index 0 with index 1, index 2 with index 3 and so on. If there is an odd number of values, the final value is not moved.

5. **[Difficulty: **]** Implement a function called `calculate_variance`(). The function takes one parameter, `numbers` (type: int[]), and returns the variance of a sample. Variance is the mean of the squares of the deviations from the arithmetic mean of a data set.

   $$variance,\ \sigma^2 = \frac{\Sigma(x_i - \bar{x})^2}{n-1}$$

variance:	σ^2
term in data set:	x_i
sample mean:	$\bar{x}$
sum:	Σ
sample size:	n

   Reference: https://www.wkihow.com/Calculate-Variance

6. **[Difficulty: **]** Implement the `is_palindrome` function. A palindrome is a string that reads the same forward or reverse. The program should only consider the uppercase and lowercase alphabets and ignore digits, empty spaces or any special characters in the input string when evaluating for a palindrome.

7. **[Difficulty: *]** Implement a function `count_num_vowels` that takes in a parameter, `sentence`. This function will return the number of vowels (i.e. the characters `'a'`, `'e'`, `'i'`, `'o'` and `'u'`) ignoring case.

8. **[Difficulty: *]** Implement a function `reverse` that takes in a parameter, word and reverse the characters in it (first character is now the last, 2nd character is now the 2nd last ...).

9. **[Difficulty: *]** Implement your own function `my_strlen` that returns the length (i.e., the number of characters) of the string without using the strlen function (without including `<string.h>`). **Hint:** look for '\0'.

10. **[Difficulty: **]** Implement your own **`my_strcmp`** function. This function is used to compare two strings two strings str1 and str2 . If two strings are the same then strcmp() returns 0. Otherwise, it returns a non-zero value. This function compares strings character by character using the ASCII value of the characters. For example 'abc' and 'abb' returns a value of 1 ('c' - 'b').
Note: You should not make use of `strlen` function in the `string.h` library.

11. **[Difficulty: ***]** Implement a function `reverse_words` that takes in a parameter, and reverse the word in the string. For example, "`apple and orange`" will become "`orange and apple`".

OPTIONAL

12. **[Difficulty: *]** Implement the `index_of` function. This function takes in two parameters:
    a. `numbers` (type: `int[]`) : This array contains at least one element.
    b. `n` (type: `int`): the number of elements in numbers
    This function returns the first index at which the value occurs in the array. Otherwise, it returns -1. For example, if the list contains {5, 7, 9, 2, 3, 5}, the first index of the value 5 is 0.

13. **[Difficulty: *]** Implement the `last_index_of` function. This function takes in two parameters:
    a. `numbers` (type: `int[]`) : This array contains at least one element.
    b. `n` (type: `int`): the number of elements in numbers
    This function returns the last index at which the value occurs in the array. Otherwise, it returns -1. For example, if the list contains {6, 7, 9, 2, 3, 6}, the last index of the value 6 is 5.

14. **[Difficulty: *]** Implement the `range` function. This function takes in two parameters:
    a. `numbers` (type: `int[]`) : This array contains at least one element, and is **sorted**.
    b. `n` (type: `int`): the number of elements in numbers
    This function returns the range of values (largest value - smallest value) that occurs in the array. For example, if the list contains {5, 7, 19, 2, 3, 5}, the range is 17 (19 - 2). If there is one element (e.g. {31}), this function returns 1. If the largest and smallest values are the same (e.g. {31, 31}), then the range is 1.

15. **[Difficulty: *]** Implement the `mode` function. This function takes in two parameters:
    a. `numbers` (type: `int[]`) : This array contains at least one element, and is **sorted**.
    b. `n` (type: `int`): the number of elements in numbers
    This function returns the most frequently occurring element in the array. If there is a tie, pick the number with the lower value. For example, if the array passed contains the values {3, 5, 5, 7, 7}, your method should return 5 (Both 5 & 7 occurs 2 times, and 5 is the smaller value).