Week 10: In-Class Exercises

Please download the resource from *http://blue.smu.edu.sg/cs101/ice10-resource.zip*

1. **[Difficulty: *]** Implement the `get_gradient` function. The function takes in 2 parameters:
   a. `p1` (type: `point`)
   b. `p2` (type: `point`)

   The gradient is calculated using the formula:

   $$gradient = \frac{p1.y - p2.y}{p1.x - p2.x}$$

2. **[Difficulty: *]** Implement the `get_oldest_person` function. This function returns the oldest person regardless of age. If there is a tie for ages, print the one with the **SMALLER** index. Assume the array has at least 1 element.

3. **[Difficulty: *]** Implement the `get_oldest_male` function. This function returns the oldest person regardless of age. If there is a tie for ages, print the one with the **LARGER** index.

4. **[Difficulty: *]** Implement the `get_youngest_female` function. This function returns the youngest person regardless of age. If there is a tie for ages, print the one with the **SMALLER** index.

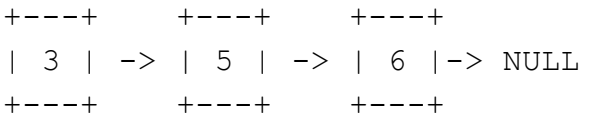
5. **[Difficulty: *]** Implement the `get_age` function. This function returns the age(current year - birth year) of the student.

6. **[Difficulty: *]** Implement the `is_equals` function. This function returns true if both the person's name and age are the same.

   Hint: Use the strcmp function in the string library.
   https://www.programiz.com/c-programming/library-function/string.h/strcmp

7. **[Difficulty: *]** A data structure is a way of organising a set of data. It defines how the data is stored, as well as the set of operations (e.g. access, add, delete, update) that can be performed on the stored data. Data structures are essential tools for programmers, as each structure has a set of benefits that make it useful for solving certain types of problems.

   A linked list is a linear data structure, in which the elements are not stored in contiguous memory locations. The elements in a linked list are linked using pointers. The principal benefit of a linked list over a conventional array is that the list elements can be easily inserted or removed without reallocation or reorganization of the entire structure.

   Implement the `size` function that calculates the number of nodes the list has.

9. **[Difficulty: **]** Given a singly linked list, return the middle element.
    a. If the list is 1 -> 2 -> 3 -> NULL, returns 2.
    b. If the list is 3 -> 4 -> 5 -> 6 -> NULL, returns 5 which is the second middle element

    Reference: https://en.wikipedia.org/wiki/Cycle_detection

10. **[Difficulty: *]** Write a function called `is_good_password`. It must return `True` if the following conditions are met:
    a. The password must be at least 6 characters long and at most 20 characters
    b. It must contain at least one lowercase letter, one uppercase letter, and one number.

    Otherwise, the function must return `False`.

11. **[Difficulty: **]** Write a function called `py_slice`. It takes in the following parameters:
    a. `orig` (type: `char[]`): This is the original string
    b. `dest` (type: `char[]`): This stores the extracted substring
    c. `start` (type: `int`): the starting position (inclusive). Assume start greater or equal to 0.
    d. `end` (type: `int`): the ending position (exclusive). Assume end greater or equal to 0.
    e. `step` (type: `int`): specify the "steps" to take from start to end index. If step is negative, the string is parsed backwards

OPTIONAL

12. **[Difficulty: **]** Write a function called is_anagram. Two words are anagrams if one word can be formed from the other by rearranging the letters. For example, "listen" and "silent" are anagrams. Assume that both words are made up of lowercase characters.

13. **[Difficulty: ***]** There is a belief that humans are able to read even when the order of the letters are misplaced. Try reading this:

    I cnduo't bvleiee taht I culod uesdtannrd waht I was rdnaieg

    Reference: https://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/

    Implement **scramble_sentence** that scrambles every word in the text leaving the first and last character untouched. Assume that the text has words separated by a single space. Maintain single space between words in the scrambled version.

    For example, if the text reads "`Programming is really fun`". One version after scrambling could be "`Pgmnarirmog is rlelay fun`". Note that since we scramble only the middle letters, "is" and "fun" in the example are not scrambled.

    Hint:
    1. Use Fisher-Yates shuffle: https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle
    2. Use the `strtok` function in the `string` library. An example is given on the next page.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void) {
    char sentence[] = "C is so fun";
    char *token_ptr = strtok(sentence, " ");

    while (token_ptr != NULL) {
        printf("%s \n", token_ptr);
        token_ptr = strtok(NULL, " ");
    }

    // observes the below output. is sentence modified?
    printf("Sentence: %s \n", sentence);
}
```

14. **[Difficulty: **]** Implement the `longest_common_prefix` function. The function takes in three parameters:
    a. `s1` (type: `char[]`)
    b. `s2` (type: `char[]`)
    c. `s3` (type: `char[]`)

    The function finds the longest common prefix of s1 and s2, and places the result in s3.

15. **[Difficulty: **]** Implement `longest_common_suffix` function. The function takes in three parameters:
    a. `s1` (type: `char[]`)
    b. `s2` (type: `char[]`)
    c. `s3` (type: `char[]`)

    The function finds the longest common suffix of s1 and s2, and places the result in s3.
    Note: Do not use any function in the string library.