ICE 12: Trial LT2

Resource: http://blue.smu.edu.sg/cs101/ice12-resource.zip

1. **[Difficulty: *]** Implement the `count_high_temperatures()` function. This function takes in two parameters:
   a. `inputs` (type: `double[]`): a list of numbers representing a sequence of body temperature readings.
   b. `n` (type: `int`): the number of elements in the array named `inputs`.

   The function returns the number of temperature readings in the input list that are strictly greater than 37.5.

2. **[Difficulty: **]** Implement the `largest_column_first` function. This function takes in 3 parameters:
   a. `n` (type: `int`): The number of rows for the `matrix`.
   b. `m` (type: `int`): The number of columns for the `matrix`.
   c. `matrix` (type: `int[][cols]`): The matrix with `n` rows and `m` columns.

   This function will find the column with the largest sum, and switch it with the first column in the `matrix`. For example, if the matrix is

   2 4 9
   1 5 8
   3 6 7

   The first column (in bold) has a sum of 6 (2 + 1 + 3), the second column has a sum of 15 (4 + 5 + 6), and the third column has a sum of 24 (9 + 8 + 7). The values in the first column (sum: 6) will be swapped with the values in the last column(sum: 24). The matrix after this function call will be as follows:

   4 9 2
   5 8 1
   6 7 3

3. **[Difficulty: **]** Implement the `identity` function. This function takes in 2 parameters:
   a. `n` (type: `int`): The number of columns and rows for the `matrix`. n is at least 1.
   b. `matrix` (type: `int[][n]`): An initialized 2-dimensional square matrix of size n.

   This function will initialize the `matrix` with ones on the main diagonal and zeros elsewhere. For example, a 3 x 3 identity matrix is as follows:

   1 0 0
   0 1 0
   0 0 1

4. **[Difficulty: **]** Implement the `rotate()` function. This function takes in 3 parameters:
    a. arr (type: int[]): The integer array
    b. size (type: int): The size of the array
    c. n (type: int): the number of steps to rotate (anti-clockwise).

    This function rotates the int array by n steps. if the array is

1	2	3	4	5	6	7

Rotating 2 steps clockwise will result in:

3	4	5	6	7	1	2

(Note: the first 2 numbers are now at the back of the array)

5. **[Difficulty: ***]** Implement the `get_email()` function. This function takes in one parameter, `sentence` (type: `char[]`) and returns the first email address located in the sentence. You can assume that the `sentence` has no punctuation and each word is separated by at least one whitespace.

6. **[Difficulty: *]** Implement the `get_repeated_lowercase_letters()` function. This function returns all the lowercase characters (case-sensitive) that are repeated(more than one occurrence) in a string in sorted ascending order (`'a' < 'b' < .. < 'z'`). If there are no repeated characters, this function returns an empty string.

OPTIONAL

7. **[Difficulty: *]** Implement the **`print_dancing_string`** function. This function takes two parameters:
    1. **`sentence`** (type: `char *`)
    2. **`start`** (type: `char`)

    This function ***prints out*** all the characters in the `sentence` in three rows in a zigzag pattern, enclosed by vertical bars on both sides, as shown by the examples below. Depending on the value of **`start`**, the zigzag pattern starts at different positions.

    If the **`start`** is `'T'`, then the zigzag pattern starts from the top row.
    If the **`start`** is `'M'`, then the zigzag pattern starts from the middle row and moves upward first.
    If the **`start`** is `'B'`, then the zigzag pattern starts from the bottom row.
    You can assume that the value of **`start`** is always `'T'`, `'M'` or `'B'`.

E.g. 1: If the function is invoked like this:

print_dancing_string('a', 'T')

the statement generates the following output:

```
|a|
| |
| |
```

E.g. 2: If the function is invoked like this:

print_dancing_string('abcdefghi', 'T')

the statement generates the following output:

```
|a   e   i|
| b d f h |
|  c   g  |
```

E.g. 3: If the function is invoked like this:

print_dancing_string('abcdefghi', 'M')

the statement generates the following output:

```
| b   f   |
|a c e g i|
|   d   h |
```

E.g. 4: If the function is invoked like this:

print_dancing_string('abcdefghi', 'B')

the statement generates the following output:

```
|  c   g  |
| b d f h |
|a   e   i|
```

E.g. 5: If the function is invoked like this:

print_dancing_string('', 'T')

the statement generates the following output:

```
||
||
||
```

8. **[Difficulty: **]** Implement the `get_highest_occurring_character()` function. This function takes in one parameter, `sentence` (type: `char[]`), and returns the character (case-insensitive) with the highest occurrence of the character in the `sentence`. If the string is empty, return '\0'.

9. **[Difficulty: *]** Implement the `get_longest_word` function. This function takes 2 parameters:
   a. word_array: This is the array of words.
   b. n: The number of elements in `word_array`.

   This function returns a pointer to the longest word.

10. **[Difficulty: *]** Implement the `count_names_with_space` function. It takes in 2 parameters:
    a. names (type: char *[]): An array of strings (char array).
    b. n (type: int): the number of strings in `names`.

11. **[Difficulty: **]** Write a program that prompts the user for a valid smu email address. You can assume that the user enters a single word (a string without any space). Keep prompting the user until the input string contains the substring "@smu.edu.sg" at the end and no other '@' anywhere else. Also make sure that there is a username portion prefixing the substring @smu.edu.sg
    A sample run of the program looks as follows. Text in bold font is user input.

```
Please enter your SMU email address:abc@gmail.com
Invalid!
Please enter a valid SMU email address:www@sis.smu.edu.sg
Invalid!
Please enter a valid SMU email address:ma@jack@smu.edu.sg
Invalid!
Please enter a valid SMU email address:@smu.edu.sg
Invalid!
Please enter a valid SMU email address:jack@smu.edu.sg
Thanks!
```

12. **[Difficulty: ***]** Implement the `total` function. The function takes in a char array of the format `"num1 num2 … numN"` and returns the sum of all the numbers. You can assume the numbers are positive and each number will not exceed 5 digits (i.e. 99999 is the max value).

For example, `total("1 2 3")` should return 6.

Hint:

```
#include <stdio.h>

int main(void) {
    char *string = "123";
    int value;
    sscanf(string, "%d", &value);
    printf("The int value is %d\n", value);
}
```