CMSC 21 Lec11 – Pointers and Multidimensional Arrays

Instructions: Upload a pdf containing a screenshot of your code, sample outputs, comments, and explanation of the code. Also include in the pdf the github link of this assignment.

1. The program below tests whether two words are anagrams (permutations of the same letters):

```c
#include <stdio.h>
#include <ctype.h> /* toupper, isalpha */

int main(void) {

    int i,
        same = 1,
        letters[26] = {0};
    char c;

    printf("Enter first word: ");
    while ((c = getchar()) != '\n') {
        if (isalpha(c)){
            letters[toupper(c) - 'A']++;
        }
    }
    printf("Enter second word: ");
    while ((c = getchar()) != '\n') {
        if (isalpha(c)){
            letters[toupper(c) - 'A']--;
        }
    }

    for (i = 0; i < 26; i++) {
        if (letters[i] != 0) {
            same = 0;
            break;
        }
    }
    if (same) {
        printf("The words are anagrams.\n");
        return 0;
    }
    printf("The words are not anagrams.\n");
    return 0;
}
```

Enter first word: smartest

```
Enter second word: mattress
The words are anagrams.

Enter the first word: dumbest
Enter the second word: stumble
The words are not anagrams.
```

The loop in lines 12-16 reads the first word, character by character, using an array of 26 integers to keep track of how many times each letter has been seen. (For example, after the word smartest has been read, the array should contain the values 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 2 2 0 0 0 0 0 0, reflecting the fact that smartest contains one a, one e, one m, one r, twos's and two t's.

The loop on lines 17-22 reads the second word, except this time decrementing the corresponding array element as each letter is read.

Both loops should ignore any characters that aren't letters and the function islpha is used. Both should treat upper-case letters in the same way as lower-case letters. One way to do this is by using toupper() to convert all letters to uppercase.

Header <ctype.h> allows the use of functions isalpha, tolower, or toupper.

After the second word has been read, use a third loop to check whether all the elements in the array are zero. If so the words are anagrams. Hint: You may wish to use.

The issue with code:

Duplications. Lines 11-16 and Lines 17-22 are basically doing the same thing. You could write a function that accepts a word.

Your task:

- Modify the anagram code above such that following functions are added:
    - void scan_word(int occurrences[26]);
    - bool is_anagram(int occurrences1[26], int occurrences2[26]);

main will **scan_word** twice, once for each of the two words entered by the user. As each character/letter of the word is being scanned, **scan_word** will use the characters in the word to update the occurrences array.

An array for each word will be declared.

int occurrences1[26] – keep track how many times each letter occurs in word 1

int occurrences2[26] – keep track how many times each letter occurs in word 2

main will then call **is_anagram**, passing it the two arrays (occurrences1 and occurrences2), **is_anagram** will return true if the elements in the two words are identical (including that the words are anagrams) and false otherwise.

2. Convert your source code in Application Item #1 such that you operate on the arrays using pointers.