Quiz 3 (7 points) - L05(27) Tu

Date: Feb 25, 2025
Time limit: 50 minutes
Name: **Student Number**:

The Tribonacci sequence T_n is defined as follows:

$$T_0 = 0,\ T_1 = 1,\ T_2 = 1, \quad and \quad T_n = T_{n-1} + T_{n-2} + T_{n-3} \quad for \quad n \geq 3$$

- *Example 1:*
  - Inputs: `n = 5`
  - Output: `output = 7`
  - Explanation:

$$T_3 = 1 + 1 + 0 = 2$$
$$T_4 = 2 + 1 + 1 = 4$$
$$T_5 = 4 + 2 + 1 = 7$$

- *Example 2:*
  - Inputs: `n = 40`
  - Output: `output = 12960201916`

1. Implement `tribonacci()` logic (4.5 points).
2. Implement `main()` to initialize a test case, call `tribonacci()` and print results for a test case (2.5 points).

Constraints:

- The answer is guaranteed to fit within a 64-bit integer, i.e., answer $\leq 2^{63} - 1$.
- `tribonacci()` must be a recursive function, otherwise -1 point deduction.
- No warnings or/and errors.

Quiz 3 (7 points) - L06(21) Mo

Date: Feb 24, 2025
Time limit: 50 minutes
Name: **Student Number**:

You are given an array `prices` where each element `prices[i]` represents the historical price of a stock on the i-th day. Your goal is to find the maximum profit you could have by selecting one day to buy the stock and a later day to sell it. Write a function that returns the maximum profit you could earn from this transaction. If no profit can be made, return `0`.

- *Example 1:*
  - Inputs: `prices = [8, 1, 6, 4, 7, 4]`
  - Output: `6`
  - Explanation: Buy the stock on day 2 ($price = 1$) and sell on day 5 ($price = 7$), resulting in a profit of $7 - 1 = 6$. Note that you must buy the stock before you sell it, so buying on day 2 and selling on day 1 is not valid.
- *Example 2:*
  - Inputs: `prices = [8, 7, 5, 2, 1]`
  - Output: `0`
  - Explanation: In this case, no transaction results in a profit, so the maximum profit is 0.

1. Implement `maxProfit()` logic (4.5 points).
2. Implement `main()` to initialize a test case, call `maxProfit()` and print the returned value (2.5 points).

Constraints:

- The efficiency of your algorithm or time complexity does **not** matter.
- Use `double` precision for `prices`.
- Use `maxProfit(int size, int prices[size])` format and `sizeof` function in `main()` to compute the `size` of an array.
- Return `0` if `size<2`.
- No warnings or/and errors.

Quiz 3 (7 points) - L07(20) Tu

Date: Feb 25, 2025
Time limit: 50 minutes
Name: **Student Number**:

An anagram is a word or phrase formed by rearranging the letters of another word or phrase, using all the original letters exactly once. You are given two strings, `s` and `t`. Write a function that checks whether `t` is an anagram of `s`. The function should return `true` if they are anagrams, and `false` if they are not.

- *Example 1:*
  - Inputs: `s = "listen"`, `t = "silent"`
  - Output: `true`
- *Example 2:*
  - Inputs: `s = "bob"`, `t = "rob"`
  - Output: `false`

1. Implement `isAnagram()` logic (4.5 points).
2. Implement `main()` to initialize a test case, call `isAnagram()` and print the result (2.5 points).

Constraints:

- The efficiency of your algorithm or time complexity does **not** matter.
- Both strings consist only of lowercase English letters.
- Use the following format for `isAnagram()` function:

```
bool isAnagram(char s[], char t[]) {
 // Implement the logic
 // Tip: You can use strlen() to get the size of a string.
}
```

- No warnings or/and errors.

Quiz 3 (7 points) - L08(27) Mo

Date: Feb 24, 2025
Time limit: 50 minutes
Name: **Student Number**:

Given an array of integers `numbers`, return `true` if there are two numbers such that they add up to a specific `target` number. If there is no combination of two values such that their sum equals the `target`, then return `false`. **You may not use the same element twice**.

- *Example 1:*
  - Inputs to `twoSum()` function: `numbers = [2,11,7,15]`, `target = 18`
  - Output: `true`
  - Explanation: The sum of 7 and 11 is 18.
- *Example 2:*
  - Inputs to `twoSum()` function: `numbers = [-3,-1]`, `target = -2`
  - Output: `false`
  - Explanation: There is not a set of two different numbers that their sum would be equal to `-2`.

1. Implement `twoSum()` logic (4.5 points).
2. Implement `main()` to initialize a test case, call `twoSum()` and print the returned value (2.5 points).

Constraints:

- $2 \leq$ the length of array numbers $\leq 3 \times 10^4$
- The data type 32-bits integer is acceptable for both `numbers` and `target`.
- Use `twoSum(int target, int size, int numbers[size])` format and `sizeof` function in `main()` to compute the `size` of an array.
- Assume always `size>0`.
- No warnings or/and errors.