Sentient Coding Tasks

Introduction

As part of this coding exercise, you'll be provided with the following set of files, intended to be used in a series of programming tasks.

Filename	For use in…
`SentientTasks.h`	Task 1, Task 2, and Task 4
`SentientTasks.cpp`	Task 1, Task 2, and Task 4
`ProjectWorkAbstract.h`	Task 3A and Task 3B
`ProjectWork.h`	Task 3A
`ProjectWork.cpp`	Task 3A
`ProjectWorkProgress.h`	Task 3B
`ProjectWorkProgress.cpp`	Task 3B

A namespace entitled `YourNameHere` will be present throughout these files (except in `ProjectWorkAbstract.h`) which we ask you to replace as part of the exercise. For example, the Candidate Carl Coder should replace `YourNameHere` with `CarlCoder`.

The files will contain guiding comments as to which parts are intended to be modified, and which are intended to be fixed. If something is unclear, please don't hesitate to ask.

When you're finished with all of the tasks, please return the complete set of files.

Task 1

To solve this task, you are to complete the function titled `Task1_ConsecutiveHighestSum` located in `SentientTasks.cpp`.

The function should, given a vector A consisting of N integers, search the vector to find the sequence of L consecutive integer values that produce the highest total sum.

The function should return the index of the first value in the identified sequence, or 0 if N < L.

For example, given A = {10, 20, 30, 40, 5} and L = 2, the function should return 2 (since the highest sum of L=2 consecutive values is 70, which occurs at index 2).

Another example is, given A = {100, 80} and L = 1, the function should return 0 (since the highest sum of L=1 consecutive value(s) is 100, which occurs at index 0).

Assume that:

- N and L are integers within the range [0 ... 100,000]
- Each element of vector A is an integer within the range [-2,147,483,648 ... 2,147,483,647]

Complexity:

- Maximum time complexity is O(N)
- Maximum space complexity is O(N)

Task 2: Code Optimisation

To solve this task, you are to refactor the function titled `Task2_AdjustBrightness` located in `SentientTasks.cpp`.

Review the code and identify any ways to **improve** the **runtime efficiency** of the provided function, **without changing its final output**.

Assume the data in the image buffer is standard RGB8 format (by row where an unsigned char represents a single channel value).

Task 3 - Part A

To solve part A of this task, you'll be working with a class entitled `ProjectWork`, whose structure and implementation is provided in the `ProjectWork.h` and `ProjectWork.cpp` files.

The class is intended to represent a body of work in a project management software. Each body of work has a title, a color, and a start and end date. Each `ProjectWork` object may have one or more `ProjectWork` object as children.

Review the code in the files noted above and **fix all logic or coding errors** you may find.

Task 3 - Part B

To solve part B of this task, you'll be working with a class entitled `ProjectWorkProgress`, whose partial structure and partial implementation is provided in the `ProjectWorkProgress.h` and `ProjectWorkProgress.cpp` files.

A new feature is being added to the project management software that allows the user to specify their progress in completing work. The user would specify this value as a percentage, with 0.0 indicating 0%, i.e. that the task has not yet started; and 1.0 indicating 100%, i.e. that the task has been completed.

For `ProjectWorkProgress` objects, with further `ProjectWorkProgress` objects attached as children, their percentage completion is determined by the percentage completion of their children. This percentage is weighted by the duration of their children, as per the examples below:

Example 1

- Task 1 has two children, as follows:
    - Task A: 10 days duration, 100% complete
    - Task B: 90 days duration, 0% complete
- Therefore, Task 1's percentage complete is 10%
    - Calculation: (10 * 1.0 + 90 * 0.0) / (10 + 90) = 10 / 100 = 0.10

Example 2

- Task 2 has three children, as follows:
    - Task A: 40 days duration, 50% complete
    - Task B: 60 days duration, 10% complete
    - Task C: 100 days duration, 20% complete
- Therefore, Task 2's percentage complete is: 23%
    - Calculation: (40 * 0.5 + 60 * 0.1 + 100 * 0.2) / (40 + 60 + 100) = 46 / 200 = 0.23

Note that tasks can be nested any number of times, for example:

Example 3

- Task 3 has three children, as follows:
    - Task A: 20 days duration, 90% complete
    - Task B lasts for 30 days and has two children:
        - Task i: 10 days duration, 50% complete
        - Task j: 20 days duration, 50% complete
    - Therefore, Task B's percentage complete is: 50%
        - Calculation: (10 * 0.5 + 20 * 0.5) / (10 + 20) = 15 / 30 = 0.50
- Therefore, Task 3's percentage complete is: 66%
    - Calculation: (20 * 0.9 + 30 * 0.5) / (20 + 30) = 33 / 50 = 0.66

Example 4

- Task 4 has no children. It has a duration of 6 days and is 40% complete.

Using the `ProjectWorkProgress` class, partially implemented in `ProjectWorkProgress.h` and `ProjectWorkProgress.cpp` as a starting point, update the class to accommodate this new feature.

List any assumptions or considerations you made in determining its design.

Task 4

To solve this task, you are to complete the function titled `Task4_DownscaleByMax` located in `SentientTasks.cpp`.

The function should accept an existing image as input and downscale it by a factor of 2 (the input image width and height are double the output image width and height). You should downscale the image by taking the **maximum** pixel value in each 2x2 block.

The image buffers contain raw grayscale image data (stored as in Exercise 2), where each pixel is represented by a single unsigned char value between 0 (black) and 255 (white).