20.04.2025

BLG102E

Introduction to Scientific and Engineering Computation (C)

Programming Practice Questions 3

For the first three questions, for your own convenience, you can initialize the arrays with random numbers (or even pre-determined numbers) instead of getting them from the user.

Question 1

Finding the central tendency of data isn't always a straightforward task. Measurement errors can introduce undesirable outliers, which may distort naive mean calculation. To lessen their effect, one option is to use the median instead of the mean. A more balanced alternative is to compute the *trimmed mean (*aka *truncated mean)*, which combines the benefits of both approaches. The figure below illustrates this concept. For example, 20% trimmed mean discards the top 20% and bottom 20% of the data, and then takes the average.

Write a program that calculates the p% trimmed mean of a given array, where p is also a parameter.

Question 2

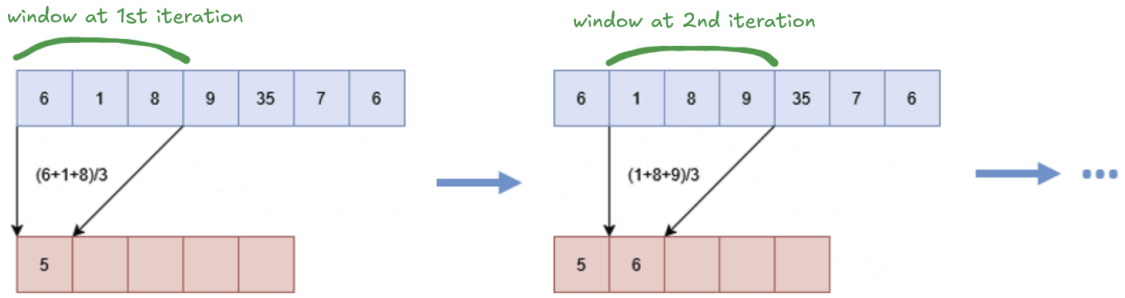
Given two **sorted** arrays of lengths N and M, create a new array of length N+M and merge the two arrays into a single sorted array. **Do not** use any sorting for this question!

Question 3

Given a list of integers and an integer k, write a function that returns `True` if there are two distinct indices i and j in the list such that `nums[i] == nums[j]` and `abs(i - j) ≤ k`

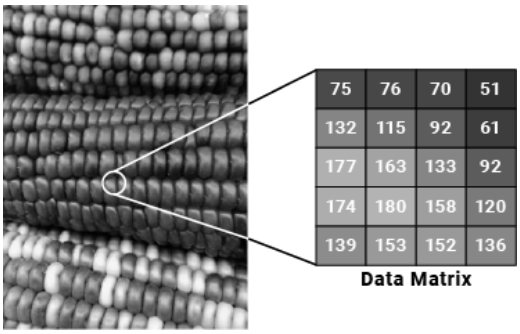
Question 4

Remember the moving average example from the recitation. In that example, we had a one-dimensional (1D) input array, and we computed the local average by sliding a window across it to produce a smoothed version of the array, as illustrated below.



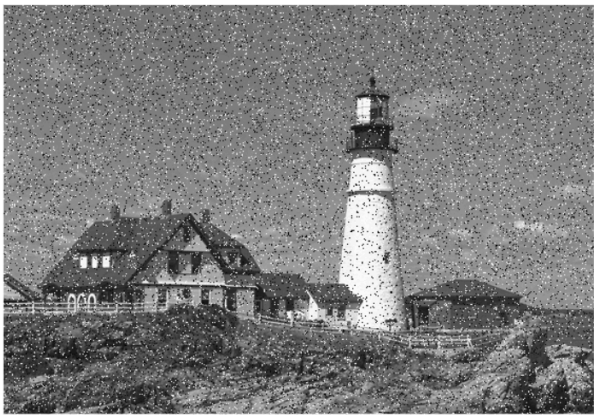
This operation helps reduce sudden changes in the data and is known as **mean filtering**. Alternatively, we could use the median value within the window instead of the mean, and this is called **median filtering**. Notice that median filtering requires sorting the values within the window.

We can extend this idea to two-dimensional (2D) arrays as well. Since a grayscale image can be represented as a 2D array, the same technique can be applied to smooth (de-noise) images.



For this task, we've provided image reading and writing functions in the `bmpio.c` file. You only need to complete the given skeleton code by implementing the **median filtering** function. You can use a window size of 3x3.

The input image and the expected output image are shown below.



Noisy image

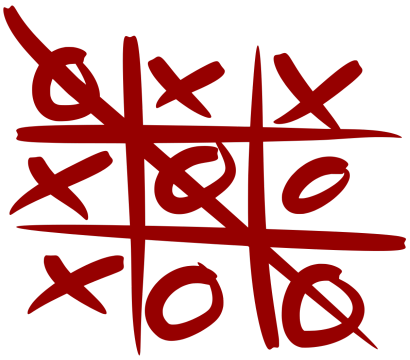
Median filtered image

To compile your code with the provided functions, use this command to compile:

```
gcc median_filtering.c bmpio.c -Wall -Werror -std=c99 -o filtering
```

From a technical perspective, this de-noising method is particularly effective for a specific type of noise known as *salt-and-pepper noise*. If you're interested, you can explore additional image de-noising techniques online.

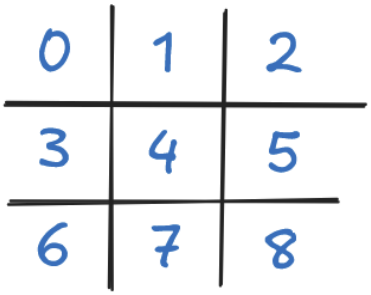
Question 5



Let's build a simple terminal-based tic-tac-toe game using **enums, arrays, and functions**. But before jumping into coding, you should first plan out your code's structure on your own. Start by writing out how you would normally play tic-tac-toe with a friend in plain, everyday language. Once you've done that, turn those steps into pseudo-code that outlines the game's logic in a structured, code-like format. Next, think about what functions you'll need to implement those steps, and how you would organize calling those functions from your main function.

If you find yourself stuck, there's a skeleton code provided as a hint, but **try not to look at it right away**. Take your time to figure out the flow and structure on your own first.

Note: During the implementation, we'll simplify the representation of the game board by using a one-dimensional array of size 9 instead of a two-dimensional 3-by-3 array. The diagram below shows how we'll map indexes 0 through 8 to positions on a 2D grid.



Question 6

Write a function to count how many times a given character appears in the 2D grid.

```
char grid[6][6] = {
        {'A', 'B', 'C', 'A', 'E', 'F'},
        {'G', 'H', 'A', 'J', 'K', 'L'},
        {'M', 'N', 'O', 'A', 'Q', 'R'},
        {'S', 'T', 'U', 'V', 'W', 'X'},
        {'Y', 'Z', 'A', 'B', 'C', 'D'},
        {'E', 'F', 'G', 'H', 'A', 'J'}
    };
```

Question 7

Imagine you're managing a warehouse where each product has:

- A **unit price** (e.g., $5.50 per item)
- A **quantity** in stock (e.g., 200 items)

Your goal is to calculate the **total value of all the inventory** in the warehouse, using arrays to store the data.

Question 8

You're building a basic student grading system. You have:

- A class of students (let’s say 6 students)
- Each student has **grades in 4 subjects**
- The data is stored in a **2D integer array**, where:
    - Each **row** represents a student
    - Each **column** represents a subject score

Your task is to **find the student with the highest total score**.

```
grades[6][4] = {
    {75, 80, 70, 85},
    {90, 95, 88, 93},
    {60, 65, 70, 55},
    {100, 90, 95, 85},
    {45, 50, 40, 60},
    {85, 80, 75, 70}
};
```

Question 9

You're given a 2D maze represented by a square matrix of size N x N where:

- `1` represents an open path,
- `0` represents a wall,
- The goal is to find **if there's a path** from the **top-left** corner `(0, 0)` to the **bottom-right** corner `(N-1, N-1)` **by only moving right or down**.

Write a recursive function to check whether such a path exists.

You are allowed to move only:

- Right: `(i, j+1)`
- Down: `(i+1, j)`

```
Example :

input

int maze[4][4] = {
    {1, 0, 0, 0},
    {1, 1, 0, 1},
    {0, 1, 1, 0},
    {0, 0, 1, 1}};
```

```
Output :

Path exists: YES
```

Question 10

You are given a secret encoded message where each number in the message represents a letter between 1 - 26 and 27 for space character (1 = A, 2 = B, ..., 26 = Z, 27=' '). You need to decode the message.

```
int encoded[] =  {6, 18, 5, 5, 27, 16, 1, 12, 5, 19, 20, 9, 14,
5};
```

About the Usage of AI Tools

ChatGPT, Claude, and many other AI tools are powerful resources that we all have actively used in our research and projects. However, especially at the beginning of the learning process, these tools can create an illusion of understanding. It might feel like you grasp the concepts when, in reality, you are only scratching the surface. Simply reading and momentarily understanding an AI-generated answer does not mean you truly comprehend the material.

To genuinely learn a new topic, particularly in this course since it is your first programming course, you need to engage with the material intentionally and put in focused effort. It is important to embrace the challenges and frustrations that come with problem-solving because this is where meaningful learning happens. AI tools can assist you along the way, but they should complement, not replace, your active participation in the learning process.

For Your Questions

Regarding this particular homework, you should ask your questions primarily to the R.A. Aycan Şahin and R.A. Enes Erdoğan.