Quiz 4 (8.5 points) - L05(27) Th

Date: Mar 20, 2025
Time limit: 55 minutes
Name: **Student Number**:

Use the following structure for `member`:

```
struct member{
 char *name;
 int SN;     // Student Number
 char *macID;
 int dob;    // Date of Birth
};
```

To create an array called `Library`, where each element is a `member`. Implement the following functions:

- `void addMember(<inputs>)`: Adds a new member to the `Library` at a given index (3.5 points).
- `void printMember(<inputs>)`: Prints the details of a member at a given index (2.5 points).
- `int main()`: where it tests the above functions. Use the following code (2.5 points):

```
int main() {

 int Size = 10;
 // CODE: Allocate memory for Library array of member struct with given Size

 char ugName[]  = "John"; // ug: User Given
 int  ugSN      = 123456;
 char ugmacID[] = "WickJ";
 int  ugdob     = 1964;

 int  InsertIndex = 2; // assume 0 <= Index < Size
 // CODE: call addMember() to insert the above member at a given InsertIndex

 int  PrintIndex = 2;  // assume 0 <= Index < Size
 // CODE: call printMember() to print an element for a given PrintIndex
}
```

Notes:

- No need to de-allocate the memory.

Quiz 4 (8.5 points) - L06(21) We

Date: Mar 19, 2025
Time limit: 55 minutes
Name: **Student Number**:

Write the following functions:

- `double **alloc2Darr(<inputs>)`: Allocate a 2D matrix with size of `nRows*nCols` (2.5 points).
- `void free2Darray(<inputs>)`: Free the memory allocated for a 2D matrix (2.5 points).
- `double **Addition(<inputs>)`: Implement the matrix addition logic (2.5 points).

We have matrix addition $C = A + B$, where:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

The resulting matrix C will also have dimensions 2×3:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix} \quad (\text{size } 2 \times 3)$$

Each element c_{ij} in C is computed as follows:

$$\begin{aligned} c_{11} &= a_{11} + b_{11} \\ c_{12} &= a_{12} + b_{12} \\ c_{13} &= a_{13} + b_{13} \\ c_{21} &= a_{21} + b_{21} \\ c_{22} &= a_{22} + b_{22} \\ c_{23} &= a_{23} + b_{23} \end{aligned}$$

Notes:

- No need to write `main()`.
- Hint: `alloc2Darr()` can be used in `Addition()` to allocate memory for C if necessary.
- Your program must be able to handle potential errors within memory allocation or matrix addition function (1 point).

Quiz 4 (8.5 points) - L07(20) Th

Date: Mar 20, 2025
Time limit: 55 minutes
Name: **Student Number**:

Write the following functions:

- `double **alloc2Darr(<inputs>)`: Allocate a 2D matrix with size of `nRows*nCols` (2.5 points).
- `double **Multiplication(<inputs>)`: Implement the matrix multiplication logic (2.5 points).
- Write `int main()` calling `allocate2Darray()` to allocate the memory for matrix A and B, and call `Multiplication()` for $C = A \times B$ (2.5 points).

We have matrix multiplication $C = A \times B$, where:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

The resulting matrix C will have dimensions 2×2:

$$C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \quad (\text{size } 2 \times 2)$$

Each element c_{ij} in C is computed as follows:

$$\begin{aligned} c_{11} &= a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \\ c_{12} &= a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ c_{21} &= a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} \\ c_{22} &= a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{aligned}$$

Notes:

- No need to de-allocate the memory.
- Hint: `alloc2Darr()` can be used in `Multiplication()` to allocate memory for C if necessary.
- Your program must be able to handle potential errors within memory allocation or matrix multiplication function (1 point).

Quiz 4 (8.5 points) - L08(27) We

Date: Mar 19, 2025
Time limit: 55 minutes
Name: **Student Number**:

Write functions to allocate (5 points) and de-allocate (3.5 points) a 3D array with the following dimensions:

- `nRows` integer: number of rows.
- `nCols` integer array: number of columns for each row; thus, `nCols[i]` represents the number of columns in the i-th row.
- `nDepth` integer: the depth of each cell in the array, defining the third dimension.

Your function should dynamically allocate memory based on these dimensions and **initialize all elements to zero**.

- *Example 1:*
  - Inputs: `nRows = 2`, `nCols = [3, 2]`, `nDepth = 4`
  - Explanation: Functions must allocate/de-allocate a 3D array with 2 rows. **Row 1** has 3 columns, each with a depth of 4 layers (depth). **Row 2** has 2 columns, each also with a depth of 4 layers.

Notes and Constraints:

- No need to write `int main()`.
- The data type is 64bits floating-point.