Array Operations Assignment: Total Points 100: Deadline 24 September 2024, 11:59 PM

Objective:

This assignment is designed to help students to implement and understand basic array ADT operations, including insertion, deletion, and updating elements in an array. This will help reinforce your knowledge of dynamic data handling and indexing in arrays

Instructions:

1. **Implement Each Function separately:**
    - Write separate functions for each operation (insert, delete, update).
    - Ensure that your functions handle edge cases *(e.g., inserting at invalid indices).*
2. **Test Your Functions:**
    - Write a main function to demonstrate and test each operation.
    - Test your functions with a variety of inputs and edge cases *(e.g., inserting at invalid indices)*.
3. **Documentation:**
    - Comment on your code to explain the logic and any assumptions made.
    - Ensure your code is clean and well organized.
4. **Submit:**
    - Submit a single .c file containing all your functions and the main function.
    - Submit it using the format: StudentName+ID.c (Example, Jimmy Walton_202320034.c)

Function 1: Array Insertion (30 points)

Task: Write a function to insert an element into a specified position in a dynamically allocated array. You need to handle resizing the array if necessary.

The insertElement function should take the following arguments:

```
int* insertElement(int* array, int* size, int position, int value);
```

Parameters:

- array: Pointer to the current array.
- size: Pointer to the current size of the array.
- position: Index at which to insert the new element (0-based).
- value: Value to be inserted.

Returns:

- Pointer to the new array with the element inserted.
- Updates the size of the array.

Requirements:

- Allocate new memory if the array needs to be resized.
- Handle invalid positions (e.g., negative index or index greater than the size).

Function 2: Array Deletion (30 points)

Task: Write a function to delete an element from a specified position in an array. Ensure that the array is resized appropriately.

The deleteElement function should take the following arguments:

```
int* deleteElement(int* array, int* size, int position);
```

Parameters:

- array: Pointer to the current array.
- size: Pointer to the current size of the array.
- position: Index of the element to be deleted (0-based).

Returns:

- Pointer to the new array with the element deleted.
- Updates the size of the array.

Requirements:

- Handle invalid positions (e.g., index out of bounds).
- Ensure elements are shifted appropriately after deletion.

Function 3: Array Update (20 points)

Task: Write a function to update an element at a specified position in the array.

The updateElement function should take the following arguments:

```
void updateElement(int* array, int size, int position, int newValue);
```

Parameters:

- array: Pointer to the current array.
- size: Current size of the array.
- position: Index of the element to be updated (0-based).
- newValue: New value to set at the specified position.

Requirements:

- Handle invalid positions (e.g., index out of bounds).
- The function should not return anything.

Function 4: Main Function (20 points)

- Demonstrates functionality of all three operations (insert, delete, update) in the main function.
- Includes diverse test cases, including edge cases and typical scenarios.