# PRACTICAL LESSON

Lec_03

1. Inserting an element into an array involves placing a new element at a specific index, shifting existing elements if needed.

- Array: `[3, 7, 12, 9]`
- Insert `5` at index `2`
- Result: `[3, 7, 5, 12, 9]`

Answer

```
function insertElement(array, element, index, size):
    if size >= array.length:
        return "Array is full"
    // Shift elements to the right from the end to the index
    for i from size-1 down to index:
        array[i + 1] = array[i]
    // Insert the element
    array[index] = element
    size = size + 1
    return array
```

2. Deleting an element from an array involves removing an element at a specific index and shifting elements to fill the gap.

- Array: `[3, 7, 5, 12, 9]`
- Delete element at index `2`
- Result: `[3, 7, 12, 9]`

```
function deleteElement(array, index, size):
    if index >= size or index < 0:
        return "Invalid index"
    // Shift elements to the left from index + 1
    for i from index to size-2:
        array[i] = array[i + 1]
    size = size - 1
    return array
```

3. Write a function to insert an element at the beginning of an array.

- Input: Array = `[4, 8, 15]`, Element = `2`
- Output: `[2, 4, 8, 15]`

4. Write a function to insert an element at the end of an array.

- Input: Array = `[4, 8, 15]`, Element = `16`
- Output: `[4, 8, 15, 16]`

5. Traversing an array using a pointer involves accessing each element by incrementing the pointer.

- Array: `[5, 10, 15, 20]`

6. Inserting an element into an array using pointers involves shifting elements by manipulating memory addresses.

- Array: `[3, 7, 12, 9]`
- Insert `5` at index `2`
- Result: `[3, 7, 5, 12, 9]`

```
function insertElement(array, element, index, size, capacity):
    if size >= capacity:
        return "Array is full"
    pointer = array + size - 1  // Point to the last element
    while pointer >= array + index:
        *(pointer + 1) = *pointer  // Shift right
        pointer = pointer - 1
    *(array + index) = element  // Insert element
    size = size + 1
    return array
```

7. Deleting an element involves shifting elements left using pointers to overwrite the deleted element.

- Array: `[3, 7, 5, 12, 9]`
- Delete element at index `2`
- Result: `[3, 7, 12, 9]`

```
function deleteElement(array, index, size):
    if index >= size or index < 0:
        return "Invalid index"
    pointer = array + index
    while pointer < array + size - 1:
        *pointer = *(pointer + 1)  // Shift left
        pointer = pointer + 1
    size = size - 1
    return array
```