a. **Given an input list and a target element, output the index of the 3rd occurrence of that element in the list. If there are not 3 occurrences, display a message. 20pt**

```
ALGORITHM Search(inputList[0...size-1], targetElement)
    // Searches for a given value in a given array
    //Input: An array inputList[0…size-1] and a targetElement;
    //Output: The index of third occurrence of targetElement

    currentIndex ← 0
    occurrence ← 0
    targetElement ← 0

    for currentIndex ← 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ← occurrence + 1
        if occurrence = 3 do
            display currentIndex
            occurrence ← occurrence + 1
        currentIndex ← currentIndex +1
    if occurrence < 3 do
        display "There are not 3 occurrences of 0 in the list."

    return 0
```

b. **Modify your algorithm from a) to output the index of the nth occurrence of the target element in the list. 20pt Changes highlighted in red**

```
ALGORITHM Search(inputList[0...size-1], targetElement)
    // Searches for a given value in a given array
    //Input: An array inputList[0…size-1] and a targetElement;
    //Output: The index of nth occurrence of targetElement

    currentIndex ← 0
    occurrence ← 0
    targetElement ← 5
    Request value of n from user

    for currentIndex ← 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ← occurrence + 1
        if occurrence = n do
            display currentIndex
            occurrence ← occurrence + 1
        currentIndex ← currentIndex +1
    if occurrence < n do
        display "There are not [n] occurrences of 0 in the list."

    return 0
```

c. **Can you make your b) algorithm more efficient? How? Write a new pseudocode or explain in your own words what strategies you would employ. 10pt Changes highlighted in red**

```
ALGORITHM Search(inputList[0…size-1], targetElement)
    // Searches for a given value in a given array
    //Input: An array inputList[0…size-1] and a targetElement;
    //Output: The index of nth occurrence of targetElement

    currentIndex ← 0
    occurrence ← 0
    targetElement ← 0
    Request value of n from user

    for currentIndex ← 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ← occurrence + 1
        if occurrence = n do
            display currentIndex
            return 0
        currentIndex ← currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."

    return 0
    /*by returning 0 once the nth occurrence of targetElement is found, we no longer continue iterating through
    the end of the list, thus, making the algorithm more efficient.*/
```

d. **Modify any of the a) or b) algorithms to output a message if the target element is found at least n times, and if yes, print how many times it was found in the entire list. 20pt Changes highlighted in red**

```
ALGORITHM Search(inputList[0…size-1], targetElement)
    // Searches for a given value in a given array
    //Input: An array inputList[0…size-1] and a targetElement;
    //Output: The index of third occurrence of targetElement

    currentIndex ← 0
    occurrence ← 0
    targetElement ← 0
    Request value of n from user

    for currentIndex ← 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ← occurrence + 1
        if occurrence = n do
            display currentIndex
            occurrence ← occurrence + 1
        currentIndex ← currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."
    else if occurrence >= n do
        display "The number [targetElement] was found at least [n] times,
        specifically, [occurrence] times!"

    return 0
```

e. **What is the basic operation in your algorithms? Write instances of the problem for list length of 15 and n=4 that achieve best-case and worst-case scenarios for each algorithm. Specify the value of the target element. 10pt**

*****NOTE: If n = 4 for these instances, then the algorithms for (a) and (b) will be the same, as the instructions for (a) state that n = 3.**

Best Case: A & B

```
ALGORITHM Search(inputList[15], targetElement)
    // Searches for a given value in a given array
    //Input: inputList[0, 0, 0, 0,  4, 6, 2, 7, 9, 4, 7, 3, 8, 3, 2]
    //Output: The index of nth occurrence of targetElement

    currentIndex ⟵ 0
    occurrence ⟵ 0
    targetElement ⟵ 0
    n ⟵ 4

    for currentIndex ⟵ 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ⟵ occurrence + 1
        if occurrence = n do
            display currentIndex
            occurrence ⟵ occurrence + 1
        currentIndex ⟵ currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."

    return 0
```

Worst Case: A & B

```
ALGORITHM Search(inputList[15], targetElement)
    // Searches for a given value in a given array
    //Input: inputList[3, 2, 1, 4,  4, 6, 2, 7, 9, 4, 7, 3, 8, 3, 2]
    //Output: The index of nth occurrence of targetElement

    currentIndex ⟵ 0
    occurrence ⟵ 0
    targetElement ⟵ 0
    n ⟵ 4

    for currentIndex ⟵ 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ⟵ occurrence + 11
        if occurrence = n do
            display currentIndex
            occurrence ⟵ occurrence + 1
        currentIndex ⟵ currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."

    return 0
```

Best Case: C

```
ALGORITHM Search(inputList[15], targetElement)
    // Searches for a given value in a given array
    //Input: inputList[0, 0, 0, 0,  4, 6, 2, 7, 9, 4, 7, 3, 8, 3, 2]
    //Output: The index of nth occurrence of targetElement

    currentIndex ⟵ 0
    occurrence ⟵ 0
    targetElement ⟵ 0
    Request value of n from user

    for currentIndex ⟵ 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ⟵ occurrence + 1
        if occurrence = n do
            display currentIndex
            return 0
        currentIndex ⟵ currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."

    return 0
```

Worst Case: C

```
ALGORITHM Search(inputList[15], targetElement)
    // Searches for a given value in a given array
    //Input: inputList[3, 2, 1, 4,  4, 6, 2, 7, 9, 4, 7, 3, 8, 3, 2]
    //Output: The index of nth occurrence of targetElement

    currentIndex ⟵ 0
    occurrence ⟵ 0
    targetElement ⟵ 0
    Request value of n from user

    for currentIndex ⟵ 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ⟵ occurrence + 1
        if occurrence = n do
            display currentIndex
            return 0
        currentIndex ⟵ currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."

    return 0
```

Best Case: D

```
ALGORITHM Search(inputList[15], targetElement)
    // Searches for a given value in a given array
    //Input: inputList[0, 0, 0, 0,  4, 6, 2, 7, 9, 4, 7, 3, 8, 3, 2]
    //Output: The index of nth occurrence of targetElement

    currentIndex ⟵ 0
    occurrence ⟵ 0
    targetElement ⟵ 0
    Request value of n from user

    for currentIndex ⟵ 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ⟵ occurrence + 1
        if occurrence = n do
            display currentIndex
            occurrence ⟵ occurrence + 1
        currentIndex ⟵ currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."
    else if occurrence >= n do
        display “The number [targetElement] was found at least [n] times,
        specifically, [occurrence] times!”

    return 0
```

Worst Case: D

```
ALGORITHM Search(inputList[15], targetElement)
    // Searches for a given value in a given array
    //Input: inputList[3, 2, 1, 4,  4, 6, 2, 7, 9, 4, 7, 3, 8, 3, 2]
    //Output: The index of nth occurrence of targetElement
    targetElement ⟵ 0
    Request value of n from user

    for currentIndex ⟵ 0 to size do
        if inputList[currentIndex] = targetElement do
            occurrence ⟵ occurrence + 1
        if occurrence = n do
            display currentIndex
            occurrence ⟵ occurrence + 1
        currentIndex ⟵ currentIndex +1
    if occurrence != n do
        display "There are not [n] occurrences of 0 in the list."
    else if occurrence >= n do
        display “The number [targetElement] was found at least [n] times,
        specifically, [occurrence] times!”

    return 0
```

*****NOTE: Efficiency-wise, the best and worst case of (d) are the same because to find the total occurrences of a single number, you need to iterate through the entire list.**