NAME: GIKURU JOSEPH NDERITU

REG NO: SCT212-0574/2022

QUIZ 1: DATA STRUCTURES AND ALGORITHMS

YEAR:2.2

COURSE: BSc COMPUTER TECHNOLOGY

Question One (1) Remove Duplicates from Sorted Array

Given a sorted array `number`, remove the duplicates in place such that each element appears only once and returns the new length.

Do not allocate extra space for another array, you must do this by modifying the input array in place with `O(1)` extra memory.

```
#include <stdio.h>

int removeDuplicates(int* nums, int numsSize){
   if (numsSize == 0) return 0;
   int pos = 1;
   for (int i = 1; i < numsSize; i++) {
      if (nums[i] != nums[i-1]) {
         nums[pos] = nums[i];
         pos++;
      }
   }
   return pos;
}
```

CODE EXPLANATIONS.

1. It checks if the array is empty (**numsSize == 0**). If so, it returns 0 since there are no elements to process.
2. It initializes a variable **pos** to 1. This variable represents the position where the next unique element should be placed in the modified array.
3. It iterates through the array starting from index 1 (**for (int i = 1; i < numsSize; i++)**). The loop compares each element (**nums[i]**) with its previous element (**nums[i-1]**).

4. If the current element is different from the previous one, it means a new unique element is found. The code then updates the array at position **pos** with the current element and increments **pos** to point to the next position for the next unique element.
5. At the end of the loop, the modified array contains only unique elements, and the function returns the value of **pos**, which represents the new length of the array with duplicates removed

Question Two (2) - Rotate Array

Given an array, rotate the array to the right by *k* steps, where *k* is non-negative.

```
def rotate (nums, k):
    n = len(nums)
    k %= n

    # Reverse the entire array
    nums.reverse()

    # Reverse the first k elements
    nums[: k] = reversed(nums[:k])

    # Reverse the rest of the elements
    nums[k:] = reversed(nums[k:])
```

This function takes as input a list of numbers and a non-negative integer k, and it modifies the list in place to rotate it to the right by k steps.

For example, if we call rotate ([1, 2, 3, 4, 5, 6, 7], 3), the function will modify the list to become [5, 6, 7, 1, 2, 3, 4].

Question Three (3) - Contains Duplicate

Given an array of integers, find if the array contains any duplicates.

Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

```
#include <stdbool.h>
#include <stdlib.h>

int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}

bool containsDuplicate(int* nums, int numsSize) {
    qsort(nums, numsSize, sizeof(int), compare);

    for (int i = 0; i < numsSize - 1; i++) {
        if (nums[i] == nums[i + 1]) {
            return true;
        }
    }

    return false;
}
```

EXPLANATIONS

This is a C function to check if an array of integers contains any duplicates. The function first sorts the array using the **qsort** function from the standard library. It then iterates through the sorted array and checks if any two adjacent elements are equal, indicating a duplicate. If it finds any duplicates, it returns **true**; otherwise, it returns **false**.

Question Four (4) - Single Number

Given a non-empty array of integers `nums`, every element appears twice except for one. Find that single one.

```
#include <stdio.h>

int findSingle(int arr[], int n) {
    int result = 0;
    for (int i = 0; i < n; i++) {
```

```
        result ^= arr[i];
    }
    return result;
}

int main() {
    int arr[] = {4, 1, 2, 1, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("The single integer is: %d\n", findSingle(arr, n));
    return 0;
}
```

CODE EXPLANATION

The **findSingle** function initializes a variable **result** to 0 and XORs it with each element of the input array. The XOR operation cancels out duplicate integers, leaving only the single integer that does not appear twice. The function returns the value of **result**, which is the single integer that does not appear twice in the array.