Άσκηση 4.6 (1/2)

Δύο μονοδιάστατοι πίνακες, έστω `a` και `b`, είναι όμοιοι εάν περιέχουν ακριβώς τα ίδια στοιχεία, όχι απαραίτητα στις ίδιες θέσεις.
Να γραφεί μια συνάρτηση με όνομα `check_identical` με ορίσματα εισόδου δύο μονοδιάστατους πίνακες ακεραίων θετικών αριθμών με ίσιο μέγεθος `N` (`N`=γνωστό).

Η συνάρτηση `check_identical` θα επιστρέφει στη `main()` τα παρακάτω:

- την τιμή 0 εάν οι δύο πίνακες περιέχουν ακριβώς τα ίδια στοιχεία.
- την τιμή 1 εάν οι δύο πίνακες ΔΕΝ περιέχουν ακριβώς τα ίδια στοιχεία, καθώς και έναν νέο πίνακα που θα περιέχει σε αύξουσα διάταξη, (χωρίς να χρησιμοποιηθεί διαδικασία ταξινόμησης, όλα τα μη κοινά στοιχεία των δύο πινάκων).

Σημείωση: η συνάρτηση ΔΕΝ θα πρέπει να περιέχει εντολές `printf`.

Στη συνέχεια να γραφεί ένα πρόγραμμα που να:

1. γεμίζει τους πίνακες `a` και `b`, με κατάλληλη χρήση της συνάρτησης δημιουργίας τυχαίων αριθμών `rand()`, ώστε οι τιμές να ανήκουν στο διάστημα [`1, 30`]
2. καλεί τη συνάρτηση `check_identical`, και εμφανίζει τα σχετικά αποτελέσματα.

Άσκηση 4.6 (2/2)

Παράδειγμα με `N=8`:

Πίνακας `a`:

4	3	8	14	11	9	3	17

Πίνακας `b`:

14	9	7	19	17	3	9	8

Αποτελέσματα στη `main()`:

`1` (δηλαδή, οι πίνακες δεν είναι όμοιοι)

Μη κοινά στοιχεία σε αύξουσα διάταξη: `4, 7, 11, 19`

Άσκηση 4.6 - printArray


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 8

void printArray(int *arr, int size) {
   for (int i = 0; i < size; i++) {
       printf("%d ", arr[i]);
   }
   printf("\n");
}
```

Άσκηση 4.6 - getRandomNumbers


```c
int *getRandomNumbers(int size, int low, int high) {

    int *arr = (int *)malloc(size * sizeof(int));
    if (!arr) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        return NULL;
    }
    for (int i = 0; i < size; i++)
        arr[i] = rand() % (high - low + 1) + low;
    return arr;
}
```

Άσκηση 4.6 - check_identical (1/3)


```
int *check_identical(int *A, int *B, int *size){
    int freq_a[31] = {0};
    int freq_b[31] = {0};
    // Μέτρηση συχνοτήτων για τους δύο πίνακες
    for (int i = 0; i < N; i++) {
        freq_a[A[i]]++;
        freq_b[B[i]]++;
    }
    // Έλεγχος αν οι πίνακες έχουν ακριβώς τις ίδιες συχνότητες
    int identical = 1;
    for (int i = 1; i <= 30; i++) {
        if (freq_a[i] != freq_b[i]) {
            identical = 0;
            break;
        }
    }
```

Άσκηση 4.6 - check_identical (2/3)


```
*size = 0;
if (identical)
   return NULL;

for (int i = 1; i <= 30; i++) {
  if ((freq_a[i] > 0 && freq_b[i] == 0) || (freq_b[i] > 0 && freq_a[i] == 0))
      (*size)++;
}
int *diff = (int *)malloc(*size * sizeof(int));
if (diff == NULL) {
    printf("Memory allocation failed!\n");
    exit(1);
}
```

Άσκηση 4.6 - check_identical (3/3)


```
    int index = 0;
    for (int i = 1; i <= 30; i++) {
        if ((freq_a[i] > 0 && freq_b[i] == 0) || (freq_b[i] > 0 && freq_a[i] == 0)) {
            diff[index] = i;
            index++;
        }
    }
    return diff;
}
```

Άσκηση 4.6 - main


```c
int main() {
    int *A = getRandomNumbers(N, 1, 30);
    int *B = getRandomNumbers(N, 1, 30);

    printf("Πίνακας A: ");
    printArray(A, N);
    printf("Πίνακας B: ");
    printArray(B, N);

    int size = 0;

    int *Diff = check_identical(A, B, &size);
    if(size == 0)
        printf("Οι πίνακες είναι όμοιοι \n");
    else{
        printf("Μη κοινά στοιχεία: ");
        for(int i = 0; i < size; i++)
            printf("%d ", Diff[i]);
        printf("\n");
        free(Diff);
    }

    return 0;
}
```