1. labor kis ZH, 2020. március 5., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

a) Készíts függvényt, amely egy egész számokat tartalmazó tömböt és egy egész számot kap paraméterként, és logikai értéket adjon visszatérési értékként, amely igaz, ha a szám benne van a tömbben, és hamis, ha nincs. Ha a szám benne van a tömbben, akkor referencia paraméterben adja vissza az elem indexét. (5p)

b) Egészítsd ki teljes C++ programmá, amely egy, a felhasználó által megadott méretű, egész számok tárolására alkalmas dinamikus tömb minden elemét az adott elem indexe+4-re állítja be (pl. egy 3 elemű tömb tartalma: 4, 5, 6), meghívja az a) feladatban létrehozott függvényt, mellyel megnézi, hogy szerepel-e a tömbben a 8, és kiírja az elem indexét, ha igen, majd felszabadítja a tömböt. A dinamikus memória kezeléséhez a C++ operátorait használd! (5p)

max. 10p, megoldási idő: 14 perc

```
#include <stdio.h>
#include <math.h>

bool keres(int tomb[], int meret, int keresett, int & index) {
    // 1p: referencia helyes megadása
    // 1p: jó visszaadott típus (bool vagy egész)
    // méretet nem adja át: -1p
    for (int i = 0; i < meret; i++) // 1p: helyes algoritmus
        if (tomb[i] == keresett) {
            index = i; // 1p: referencia helyes használata
            return true; // 1p: helyes visszaadott érték (false vagy 0)
        }
    return false;
}

int main() {
    unsigned n;
    scanf("%d", &n);
    int *t = new int[n]; // 2p: helyes tömbfoglalás new-val. Részpontszám nincs.
    for (int i = 0; i < n; i++)
        t[i] = i + 4;
    int index; // 1p: referencia értése és helyes használata: nem jár, ha csinál felesleges ref-
erencia változót a main-ben.
    if (keres(t, n, 8, index)) {  // 1p, ha nem cifrázza a logikai érték vizsgálatát, azaz nincs
==true vagy ==1
        printf("Az elem indexe: %u");
    }
    delete[]t; // 1p: ha a delete[] operátort használja, jól
    return 0;
}
```