1. labor kis ZH, 2018. március 1., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

a) Definiálj struktúrát 2D vektor tárolására, melynek x és y koordinátája double típusú! (0p)

b) Írj függvényt, amely referenciával vesz át egy 2D vektort, és a szabványos bemeneten megadott értékekkel beállítja a koordinátákat (pl. scanf segítségével). (2p)

c) Készíts függvényt, amely egy 2D vektorokból álló tömböt vesz át, és létrehoz dinamikusan egy ugyanakkora méretű, double elemű tömböt! A függvény töltse fel az új tömböt a 2D vektorok hosszaival ($\sqrt{x^2 + y^2}$), majd adja vissza az új tömböt! A dinamikus tömböt ne a C-s, hanem a C++-os módon foglald le! (4p)

d) Írj main függvényt, amely létrehoz egy 2D vektorokból álló 100 elemű tömböt, melyet feltölt a felhasználó által a szabványos bemeneten megadott értékekkel, kiszámítja a vektorok hosszát egy dinamikus tömbbe, és ki is írja a vektorok hosszát, majd felszabadítja a lefoglalt dinamikus memóriát! A feladatot a b) és c) feladatban írt függvény felhasználásával oldd meg! (4p)

max. 10p, megoldási idő: 20 perc

```cpp
#include <cstdio>
#include <cmath>

struct v2d { // 0p, ha nem jó -1p
    double x, y;
};

void set_v2d(v2d & v) { // át tud venni referenciával: 2p
    scanf("%lg %lg", &v.x, &v.y); // 0p, ha nem jó -1p
}

double * hosszak(v2d * t, int n) { // fejléc + return, ha minden jó: 1p
    double * h = new double[n]; // 2p
    for (int i = 0; i < n; i++)
        h[i] = sqrt(t[i].x*t[i].x + t[i].y*t[i].y); // 1p
    return h;
}

int main() { // minden egyéb a main-ben: 1p
    v2d t[100]; // ha nem jó, -1p
    for (int i = 0; i < 100; i++) // ha nem jó a ciklus sehol: -1p
        set_v2d(t[i]); // hívás 1p
    double * uj = hosszak(t, 100);
    for (int i = 0; i < 100; i++)
        printf("%g\n", uj[i]);
    delete [] t; // 2p
    return 0;
}
```