Programozás Alapjai 7. ZH

30. feladatsor

Szoftverfejlesztés Tanszék

2023, Ősz

Feladat Töltsd le a bíróról a `minta.zip` állományt, majd tömörítsd ki! A `feladat.c` fájlban megtalálod a feladatok megoldás-kezdeményeit. Bővítsd ezt az alább olvasható feladatok alapján! Lehetőség szerint ellenőrizd megoldásod, majd töltsd fel a `feladat.c` fájlt a bíróra!

Kiértékelés A bíró lefordítja a programot, majd lefuttatja azt a feladat pontszámának megfelelő számú tesztesettel. Egy teszteset egy bemenet-kimenet pár, amely a megfelelő feladathoz készült. A teszteset akkor helyes, ha az adott bemenethez tartozó kimenet **minden egyes karaktere** megegyezik az előre eltárolt referencia kimenettel. *További feltételek: a program futása nem tarthat tovább 5 másodpercnél, egyszerre nem fogyaszthat többet 16 MiB memóriánál és nem történhet futási hiba (pl. illetéktelen memória hozzáférés).*

Ellenőrzés Feltöltés előtt érdemes ellenőrizni a megoldásod.

1. **Fordítás** Ellenőrizd, hogy a programod lefordul-e! A bíró a `gcc -O2 -static -o feladat feladat.c` paranccsal fordít, érdemes ezt használni. A `-Wall` kapcsoló is hasznos lehet.
2. **Példa tesztesetek** Ellenőrizd, hogy a programod helyesen működik-e! A `minta.zip` tartalmaz a bíró által futtatott tesztesetek közül feladatonként egyet-egyet. Az első feladat teszteléséhez másold a programod mellé az `ex1.be` fájlt `be.txt` néven, futtasd le a programod, majd az így kapott `ki.txt` tartalmát hasonlítsd össze az `ex1.ki` fájlban található referencia kimenettel.
3. **Extra tesztesetek** Ellenőrizd a programod működését további példák segítségével! Néhány további teszteset is elérhető, de ezek csupán ellenőrzésre használhatóak, a bíró nem futtatja őket. Ezek használatához futtasd a programod a `-t` vagy `-test` kapcsolóval, például a `./feladat -test` paranccsal. Csak az első feladat teszteléséhez futtasd a programod a `./feladat -t 1` paranccsal.

1. feladat (5 pont)
A `lejto()` függvény feladata kiszámolni azt, hogy mekkora ereszkedést kell végrehajtani a legnagyobb szintkülönbségű szakasz leküzdéséhez egy szakaszokból felépülő hegyvidéki túra során. A paraméterben kapott `terkep` tömb a túraútvonal szakaszait záró állomások tengerszint feletti magasságértékeit tartalmazza. A tömböt a `-1` érték zárja. A függvény kiszámolná, hogy mekkora a legnagyobb különbség a következő és az aktuális állomás között, ott ahol a következő állomás eléréséhez lefelé kell haladni. Vedd szemügyre a próbainputot és a hozzá tartozó kimenetet.

A függvény megvalósítása hibákat tartalmaz. Javítsd ki ezeket a hibákat.

```
int lejto(int terkep[]);
```

2. feladat (5 pont)
A feladat egy függvény megírása, amely paraméterül vár egy egész számokból álló tömböt, illetve két egész számra mutató pointert. A tömbben egész számok vannak, a tömb végét a 0 érték jelzi.
A függvény feladata, hogy meghatározza a tömbben lévő elemek közül a legnagyobb pozitív, illetve legnagyobb negatív értéket. A `maxPozitiv` pointer által mutatott érték legyen a legnagyobb pozitív szám, míg a `maxNegativ` által mutatott érték pedig a legnagyobb negatív szám, ami a tömbben előfordul. Amennyiben a tömbben egy pozitív / negatív érték sem fordul elő, akkor a pointer által mutatott érték legyen a tömb végét jelző érték.
Kódold le C nyelven a függvényt! A függvény fejlécén ne változtass! A függvény inputjai a paraméterek, outputja a visszatérési érték. A függvény nem végez IO műveleteket!

```
void statisztika(int szamok[], int* maxPozitiv, int* maxNegativ);
```