el?z? fel következ?

Tartalomjegyzék

Feladatok

Dinamikus programozás

Legnagyobb közös osztó

Implementáljuk a legnagyobb közös osztó függvényt (lnko) az euklideszi algoritm segítségével. Rekurzívan és nem-rekurzívan is!

Zárt terület kifestése

Olvassuk be a picture.txt fájlt listák listájába (minden karakter egy elem)! Írjunk egy fill(x,y) függvényt, ami ugyanazt csinálja, mint a Paint kitölt? funkciója! Az (x,y) pontból kiindulva a . helyére # jelet tesz, amíg a # jel által jelölt falba nem ütközik! A módszer rekurzív: kifestjük az (x,y) pontot, majd a szomszédait, ha azok nem # jelek. Hívjuk meg a szomszédokra (akik nem # jelek) a függvényt rekurzívan. Ha nincs kit kiszínezni, akkor álljunk meg!

```
.....................................
...########################...........
...#......................#...........
...#......................#...........
...#......................#...........
...#......................#...........
...#......................#...........
...#......................#...........
...#......................#######.....
...###..................##......#.....
...#..##..............##........#.....
...#....##..........##..........#.....
...#......##......##............#.....
...#........#####................#.....
...#........#....................#.....
```

```
...#.......##..................#.....
...#.....##....................#.....
...#...##......................#.....
...############################.....
.....................................
.....................................
.....................................
.....................................
```

Huszárok

Áll a sakktáblán egy huszár. Számoljuk ki a sakktábla minden mez?jére, hogy legkevesebb hány lépéssel tudunk eljutni oda az el?bbi huszárral! A feladat megoldásához írjunk egy függvényt, **knight(x,y)**, amelynek két bemenete a huszár táblán elfoglalt helyének koordinátái. A visszatérési érték legyen egy nyolcszor nyolcas lista! Használjunk dinamikus programozást!

Állapotgép

Billenty?k

Töltsük le az alábbi adatfile-t: raw_data.txt

A file tartalma egy rövid szöveg begépelése alatt történt billenty? lenyomásokat kódolja. Az érdekes rész az 5. sortól kezd?dik:

- Az els? szó az esemény, a számunkra érdekesek a **keydown** és **keyup** események, ezek rendre a billenty? lenyomás és felengedés.
- A következ? három szám a karakter kódja, innen a 2. (azaz a sorban 3. elem) a megbízható, használjuk ezt.
- Az igaz-hamis érték a kis / nagy bet?re vonatkozik, de mi ezzel most ne foglalkozzunk.
- Az utolsó elem az érdekes még számunkra, ez az esemény id?pontja (pontosan az 1970 január 1. óta eltelt milliszekundumok).

A feladat az, hogy úgy dolgozzuk fel ezt az adathalmazt, hogy a billenty? lenyomások és felengedések közti id?t megkapjuk. Csak egy ilyen id?sor érdekel minket, a sorrend legyen a lenyomás pillanata szerint. Például az eleje így nézne ki:

```
145 80 74 ...
```

A 145-öt az alábbi két sorból kapjuk:

```
keydown 16 16 0 true 1444121075394
keyup 16 16 0 false 1444121075539
```

majd a 80-at:

```
keydown 84 84 0 true 1444121075462
keyup 84 84 0 false 1444121075542
```

a 74-et:

```
keydown 72 72 0 false 1444121075693
keyup 72 72 0 false 1444121075767
```

Az így kapott számsort írjuk egy **kimenet.txt** file-ba!

Segítség:

Rengeteg módon megoldható a feladat, ez csak egy ötlet:

- Tároljuk a már lenyomott és felengedésre váró gombokat egy szótárban.
- Ha megérkezett egy várt gomb felengedése akkor mentsük a lenyomás id?tartamát, majd töröljük a szótárból.

Bónusz: rekonstruáljuk a beírt szöveget. Vigyázat, ez a file tartalmazza a backspace, SHIFT, stb. billenty?k lenyomását is. A billenty?kódokról itt.

Zárójelek

Adott egy sztring. Cseréljük le azokat a karaktereket $ jelre, amelyek zárójelek között vannak (a zárójeleket is beleértve)! Figyelem, a zárójelek lehetnek egymásba ágyazva is, tehát, ha a bemeneti sztring *(xc)aa(c(b))*, akkor a kimenet *$$$$aa$$$$$$* legyen!

el?z? fel következ?