TREASURE

Az Adrián egy új sziget keletkezett, amelyen kincsek találhatók. A sziget N\*N-es négyzetrácsban elhelyezkedő cellákból áll. A sorokat és az oszlopokat is 1-től N-ig sorszámozzuk, a sorokat föntről lefelé, az oszlopokat balról jobbra. Bizonyos cellák kincset tartalmaznak.

Szeretnénk tudni, hogy melyek ezek a cellák. Kérdéseket tehetsz fel tetszőleges téglalapra, amelyre válaszul megkapod, hogy a téglalap hány cellája tartalmaz kincset.

Egy kérdés költsége 1 + *N*\**N* – *S*, ha a téglalapban S cella van.

FELADAT

Írj programot, amely kideríti, hogy mely cellák tartalmaznak kincset! A cél, hogy minél kevesebb költséggel határozd meg, de nem szükséges a minimálisat elérned. A Pontozás-nál látszik, hogy milyen megoldás hány pontot ér.

PÁRBESZÉD

A programod a standard inputon kérdezhet és a válaszokat a standard outputon kapja.

* Először az *N* méretet kell beolvasnod (2 ≤ *N* ≤ 100).
* A kérdésekhez 4 egész számot kell kiírnod, egy-egy szóközzel elválasztva: *R1*, *C1*, *R2* és *C2*, ahol 1 ≤ *R1* ≤ *R2* ≤ *N* and 1 ≤ *C1* ≤ *C2* ≤ *N. Hibás kérdés esetén a programod 0 ponttal befejeződik.*
* Válaszként egy egész számot kapsz, azon (*R*, *C*) sor- és oszlopindexű cellák számát, amelyekre teljesül, hogy a kérdésben szereplő téglalapban vannak, azaz *R1* ≤ *R* ≤ *R2* és *C1* ≤ *C* ≤ C2 és az (*R*, *C*) kincset tartalmaz.
* Amikor a programod befejezi a kérdéseket, akkor egy sorba az ‘END’ szöveget kell kiírnod. A következő N sor mindegyikébe N karaktert kell kiírni. Az i-edik sor j-edik karaktere ‘1’, ha (I,j) cellában van kincs, egyébként pedig ‘0’. Az eredmény kiírását követően a programod automatikusan befejeződik.
* Minden sor kiírása után flush műveletet kell végrehajtanod, ahogy a kapott példában láthatod.

Felteheted, hogy minden kérdésre helyes választ kapsz, ami nem függ a feltett kérdések sorrendjétől, azaz előre rögzített, hogy hol van kincs.

MINTAPROGRAMOK

A mintaprogramok mindhárom programozási nyelven letölthetők a versenyrendszer ‘Tasks’ lapjáról. Ezek formailag helyes párbeszédek, de nem korrekt megoldásai a feladatnak.

PONTOZÁS

Minden teszteset 10 pontos. Ha a programod nem helyes eredményt ad, akkor 0 pontot kapsz. Különben a pontszám az alábbiak szerint függ a K költségtől:

* Ha *K* ≤ 7/16 *N*4 + *N*2, akkor a pontszámod 10 pont,
* különben ha *K* ≤ 7/16 *N*4 + 2 *N*3, akkor 8 pont,
* különben ha *K* ≤ 3/4 *N*4, akkor 4 pont,
* különben ha *K* ≤ *N*4, akkor 1 pont,
* különben 0 pont.

Az elérhető pontok 40%-ában *N* legfeljebb 20.

A megoldás helyességét a kérdésektől függetlenül ellenőrzik (akár kérdések nélkül is adhatsz megoldást).

PÉLDA

A következő példában bal oldalon vannak a kiírt kérdések és a megoldás, a jobb oldalon pedig N értéke és a kapott válaszok.

|  |  |
| --- | --- |
| output | input |
|  | 2 |
| 1 1 1 1 | 0 |
| 1 2 1 2 | 1 |
| 2 1 2 2 | 2 |
| END |  |
| 01 |  |
| 11 |  |

TESZTELÉS

Vagy lokálisan, vagy az értékelő rendszerrel tesztelhetsz. Mindkét esetben készítened kell egy, a négyzetrácsot leíró tesztfile-t. Első sorában az N értéke legyen! A következő N sor mindegyike a várt eredménynek megfelelően tartalmazza a négyzetrács sorainak leírását!

Például az előző párbeszédnek megfelelő tesztfile:

2

01

11

Az értékelő rendszernek először a forráskódot kell elküldeni (Submit lapon), majd a ‘TEST’ lapon kell beküldened a tesztfile-t. Az értékelő rendszer közli, hogy helyes-e a megoldásod és közli a költséget is.

Lokális tesztelés esetén használd a letölthető treasure\_test programot! A parancssor a következő legyen: ./treasure\_test ./my\_solution input\_file. A program közli, hogy helyes-e a megoldásod, a treasure.log file tartalmazza a kérdéseket és a válaszokat.

TRAM

A zágrábi új villamoson a helyek N sorban (1-től N-ig sorszámozva) és 2 oszlopban (1-től 2-ig sorszámozva) vannak. Az *RA* sorban, *CA* oszlopban és az *RB* sorban és *CB* oszlopban levő helyek távolsága: $\sqrt{\left(R\_{A}-R\_{B}\right)^{2}+\left(C\_{A}-C\_{B}\right)^{2}}$.

Minden új utas a lehető legmesszebb akar ülni a többiektől, azaz azt a helyet választja, amitől a legközelebbi foglalt hely a lehető legmesszebb van. Ha több ilyen hely van, akkor a legkisebb sorindexűt választja. Ha ebből Is több van, akkor közülük a legkisebb oszlopindexűt. A leszállásig minden utas a helyén marad. Ha üres a villamos, akkor a felszálló utas az (1,1) helyre ül le.

FELADAT

Írj programot, amely egy érkezés és távozás eseményeket tartalmazó sorozatra megadja, hogy az érkezők hova ülnek! A villamos kezdetben üres.

M esemény van, 1-től M-ig sorszámozva, az előfordulásuk sorrendjében. ‘E’ jelenti az érkezést, ‘L’ pedig a távozást. Távozás esetén a távozónak az érkezési eseményének a sorszáma van az L betű után.

A tesztekben minden érkezéskor van legalább egy üres hely a villamoson.

INPUT

A bemenet első sorában *N* és *M* (1 ≤ *N* ≤ 150 000, 1 ≤ *M* ≤ 30 000) van, a sorok száma és az események száma. A következő *M* sor mindegyike egy-egy eseményt ír le, a *K.* közülük a K. eseményt írja le, ami vagy az 'E’ karakter, vagy az ‘L’ karakter, amit egy szóközzel elválasztva követ egy *PK* egész szám (1 ≤ *PK* < *K*), ahol *PK* a távozó utas érkezési eseményének sorszáma. Senki sem akar kétszer leszállni.

OUTPUT

Annyi sort tartalmazzon, ahány ‘E’ típusú esemény van! A sorok az érkezés sorrendjében adják meg, hogy kit hova kell leültetni: az ülőhely sor-és oszlopindexét egy szóközzel elválasztva.

PONTOZÁS

* A tesztek 25%-ában N ≤ 150 és M ≤ 150.
* A tesztek 45%-ában N ≤ 1500 és M ≤ 1500.
* A tesztek 65%-ában N ≤ 150 000 és M ≤ 1500.

RÉSZLETES VISSZAJELZÉS

Legfeljebb 50 beküldés esetén kérhetsz részletes visszajelzést.

PÉLDÁK

|  |  |  |
| --- | --- | --- |
| input3 7EEEL 2EL 1Eoutput1 13 21 23 11 1 | input13 9EEEEEEEEEoutput1 113 27 14 210 12 23 15 16 2 | input10 9EEEEL 3EEL 6Eoutput1 110 25 27 14 22 24 1 |

Splot

Olyan parkoló-rendszert építettek az Adria partján, amely soros-párhuzamos gráf konstrukcióval írható le.

A parkoló-rendszer parkolóhelyekből és azokat összekötő kétirányú útszakaszokból áll. Bármely kettőt legfeljebb egy útszakasz köt össze. Bármely parkolóhelyen legfeljebb egy autó állhat. Foglalt helyeken nem lehet áthaladni.



1. ábra: A szabályoknak megfelelő ábrák

Minden soros-párhuzamos parkoló tartalmaz két kitüntetett helyet, ezek neve kijárat (s jelöli az ábrákon) és terminál (t jelöli az ábrákon).

A soros-párhuzamos parkoló az alábbi 3 szabály ismételt rekurzív alkalmazásával kapható. A szabály tartalmazza a leírás formáját is:

1. Az egyetlen helyet tartalmazó parkoló szabályos. Ez a kijárat és egyúttal a terminál is. Az ilyen parkolóhelyet a leírásban a kis ‘o’ betű jelöli, ha a hely üres, illetve a kis ‘x’ betű, ha a hely foglalt (azaz autó áll ott).
2. Ha *G1* és *G2* szabályos parkoló, akkor a *G-vel jelölt soros kompozíciójuk is szabályos, amelyet úgy kapunk, hogy a kijárata a* *G1* kijárata, terminálja a *G2* terminálja, és a G1 terminálja és a G2 kijárata között kétirányú útszakaszt vezetünk be. Ha *E1* és *E2* a *G1* és *G2* leírása, akkor a *G* leírása az ‘S*E1E2*#’, azaz a leírás nagy ‘S’ betűvel kezdődik, ezután jön a G1 és G2 leírása, a végét pedig a ‘#’ karakter jelzi.
3. Ha *G1* és *G2* szabályos parkoló, akkor a *G-vel jelölt párhuzamos kompozíciójuk is szabályos, amelyet úgy kapunk, hogy bevezetjük az s és t új parkolóhelyeket, amelyek közül az s lesz a G kijárata, t pedig a terminálja. Bevezetünk 4 új kétirányú útszakaszt: s-s1, s-s2, t1-t, t2-t, ahol s1 a G1-nek, s2 pedig a G2-nek a kijárata, t1 a G1-nek, t2 pedig a G2-nek a terminálja.* Ha *E1* és *E2* a *G1* és *G2* leírása, akkor a *G* leírása a‘P*Es*|*E1E2*|*Et*#’, ahol az Es az új kijárat leírása, Et az új terminál leírása (mindegyik a foglaltságnak megfelelően vagy kis ‘o’ betű vagy kis ‘x’ betű). A leírás nagy ‘P’ betűvel kezdődik, ezt követi az új kijárat, majd a ‘|’ függőleges vonal. Ezután jön a két ág leírása, amit újabb ‘|’ követ. Végül az új terminál leírása és a ‘#’ karakter következik.



2. ábra: Az első tesztesetnek megfelelő parkoló gráfja

A fenti ábra leírása: ‘Po|Px|Sxo#Soo#|o#Soo#|o#’. Megjegyzendő, hogy pontosan annyi parkolóhely van, ahány kisbetű szerepel a leírásban.

Az egyetlen kimenetet, ahol az autók a parkolót elhagyhatják, az ábrán az exit felirat jelöli.

Egy parkolóhelyen álló autó nem blokkolt, ha van olyan útvonal, amely a parkolóhelyétől a kijárathoz vezet olyan helyeken keresztül, amelyek nem foglaltak.

A fenti ábrán egyik autó helye sem blokkolt. Ha lenne egy autó a parkoló terminálján (a legjobboldalibb helyen), akkor az egy autót blokkolna (a felsőt). Ha a kijárathoz tennénk autót, akkor az minden más autót blokkolna.

FELADAT

Úgy szeretnénk autókat elhelyezni, hogy egyik sem legyen blokkolt. Lehetnek már autók a parkolóban, de ezekről tudjuk, hogy egyik sem blokkolt.

Írj programot, amely kiszámítja az autók maximális számát, amelyek elhelyezhetők úgy, hogy egyik se legyen blokkolt. Ebbe a számba beleértjük a már ott levő autók számát is. Meg kell adni továbbá a maximális számú autó egy lehetséges elhelyezését, a már ott levőket nem lehet elmozdítani.

INPUT

A bemenet egyetlen sora egy soros-párhuzamos parkoló-leírást tartalmaz, amely legalább 1 és legfeljebb 100 000 karaktert tartalmaz. A leírásban csak az alábbi karakterek lehetnek: nagy ‘P’ és ‘S’, kis ‘o’ és ‘x’, ‘#’ (ASCII 35) and ‘|’ (ASCII 124) karakter. Ebben biztos nincs blokkolt autó.

Output

A kimenet 2 sorból áll. Az első sorba az M számot kell írni, az autók maximális számát, amelyek elhelyezhetők úgy, hogy egyik se legyen blokkolt.

A második sorba a maximális számú autó egy lehetséges elhelyezését kell írni. amelyben pontosan M darab kis ‘x’ kell legyen, továbbá az új ‘x’-et csak valamely bemenetbeli ‘o’ helyére tehetsz.

Több megoldás esetén bármelyik megadható.

PONTOZÁS

* Ha csak az első sor helyes, akkor a maximális pontszám 80%-át kapod.
* 30 pont szerezhető olyan bemenetekre, ahol a parkolóhelyek száma legfeljebb 20.
* További 40 pont szerezhető olyan bemenetekre, amelyekben nincs foglalt hely, azaz a bemenetben nincs ‘x’ betű.

PÉLDÁK

|  |  |
| --- | --- |
| inputPo|Px|Sxo#Soo#|o#Soo#|o#output3Po|Px|Sxo#Sox#|o#Soo#|o# | inputPo|SPo|oo|o#Px|oo|o##Po|Sxo#Po|ox|o#|o#|o#output7Po|SPo|xx|o#Px|ox|o##Po|Sxx#Po|ox|o#|o#|o# |

TESZTELÉS

A splot\_tool (amit letölthetsz az értékelő rendszerből) SVG képet készít egy szabályos leírásból, amit böngészővel nézhetsz meg. Használata:

$ ./splot\_tool splot.dummy.out.1

splot.dummy.out.1 parsed (10 parking spaces).

splot.dummy.out.1.svg created.

$ chromium splot.dummy.out.1.svg

A program ellenőrzi, hogy szabályos-e a megadott leírás, de más ellenőrzést nem csinál. Legfeljebb 200 karakteres lehet a leírás.