Programozás Alapjai 8. házi feladat

1. feladatsor

Szoftverfejlesztés Tanszék

2023, Ősz

Általános információk

A programot C nyelven kell megírni, és a *Bíró* webes felületén keresztül lehet benyújtani. Egy C program kiterjesztése `c`. A *Bíró* a fájl nevében található első pont utáni részt tekinti kiterjesztésnek.

Kiértékelés

A programot a *Bíró* fogja kiértékelni. Feltöltés után a *Bíró* a programot a `gcc` fordítóval és a `-O2 -static -o feladat feladat.c` paraméterezéssel lefordítja, majd a programot különböző tesztesetekre futtatja. Minden helyes teszteset 1 pontot ér. A teszteset akkor helyes, ha a program futása nem tartott tovább 5 másodpercnél, a futása hiba nélkül (0 hibakóddal) fejeződött be és az adott inputhoz tartozó kimenet **minden egyes karaktere** megegyezik az előre eltárolt referencia megoldással.

A *Bíró* által a `riport.txt`-ben visszaadott lehetséges hibakódok:

`Futási hiba: 6`	Memória- vagy időkorlát túllépés.
`Futási hiba: 8`	Lebegőpontos hiba, például nullával való osztás.
`Futási hiba: 11`	Memória-hozzáférési probléma, pl. tömb-túlindexelés, null pointer használat.

Minden programra vonatkozó követelmények

A program bemenő adatait a `be.txt` nevű fájlból kell beolvasni, az eredményt pedig a `ki.txt` nevű fájlba kell írni akkor is, ha ez nincs külön megemlítve a feladat leírásában. A `be.txt` állomány csak olvasásra, a `ki.txt` állomány pedig csak írásra nyitható meg, más megnyitási mód esetén a *Bíró* nem engedélyezi a hozzáférést. Más fájl megnyitását a *Bíró* szintén nem engedélyezi.

A program bemenet/kimenet leírásokban a „sor” egy olyan karaktersorozatot jelöl, amelyben pontosan egy sorvége jel (’`\n`’) található, és az az utolsó karakter. Tehát minden sort sorvége jel zár! Elképzelhető olyan output, amelyben nincs sorvége jel, de akkor a feladat kiírásának egyértelműen jeleznie kell, hogy a sorvége jel hiányzik!

A hibakód nélküli befejezést a `main` függvény végén végrehajtott `return 0;` utasítás biztosíthatja.

1. feladat (8 pont)

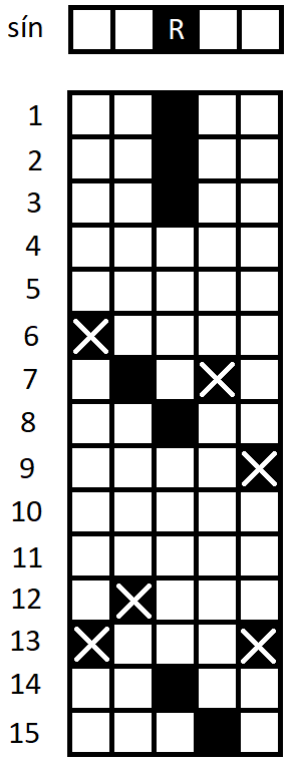
A homokviharok visszatérése nem az egyetlen időjárási tényező, amely a Mars kolonizálását fenyegeti. Mivel a Mars légköre nagyon ritka, ezért az űrből érkező meteorzáporok és űrszemét nem ég el a légkörbe való belépéskor és komoly károkat okozhat. Szerencsére erre a körülményre már fel van készülve a kolónia, elődöd, Dr. Halliburton korábban épített egy robotot, amely képes elkapni a közvetlen veszélyt okozó zuhanó tárgyakat.

Sajnos azonban a radarok felfedezték, hogy egy minden eddiginél nagyobb meteorraj tart a kolónia felé. A probléma, hogy ilyen nagy mennyiségű meteortól a robot sem tudja egymaga megvédeni a kolóniát, ezért rakétákkal kell kilőni azokat a meteorokat, amelyek átjutnának a robot által képzett védelmi vonalon. Azonban ezeknek a védelmi rakétáknak a megépítése nagy mennyiségű időt és nyersanyagot von el az épületek megerősítésétől és az egyéb felkészüléstől. A kolónia jövője tehát ismét rajtad múlik, mint informatikus neked kell meghatároznod, hogy hány rakétára lesz szükség, azaz hány meteor jutna át a robot védelmi rendszerén.

A feladatot komplikálja, hogy a Dr. Halliburton jelszava nélkül a robot forráskódjához nem férsz hozzá. Azt azonban tudod, hogy a robot intelligenciája kezdetleges, sőt elődöd feljegyzéseiből a viselkedését is teljes mértékben ismered. A robot vízszintesen mozog balra vagy jobbra egy sín mentén, és megsemmisíti a beérkező meteorokat. A meteorzápor érkezését egyenletes időszeletekre osztjuk fel. A robot szenzorai mindig csak az aktuális időszeletben érkező meteor helyzetét érzékelik. A robot időszeletenként egyet tud lépni. A robot a mozgását az alábbi szabályok szerint végzi:

- A legelső időszeletben még van ideje kiválasztani a megfelelő pozíciót, így balról jobbra az első meteort garantáltan elkapja, tulajdonképpen ez a kiindulási pontja.
- Amennyiben egy lépéssel meg tud semmisíteni egy meteort az adott időszeletben, akkor ezt a lépést megteszi. Ha azonban nem csak egy meteort tudna így megsemmisíteni az adott időszeletben, akkor döntésképtelen, így nem mozdul, akkor sem, ha ezzel akár egyet sem kap el.
- Amennyiben a következő lépésével nem tudna egy meteort sem elkapni, arra mozdul, amerre az adott időszeletben több meteort lát.
- Minden egyéb esetben mozdulatlan marad.

Egy példa a robot mozgására:



A képen az R jelzi a robot kezdeti pozícióját a sínen, ez az első időszeletnek, azaz kapott sornak felel meg (a képen 1-től indexeltünk). A feketére színezett négyzetek jelzik hol érkezik meteor. Az X-szel jelölt

fekete mezők azok a meteorok, amelyeket a robot nem tud elkapni. A helyes rakétaszám ebben az esetben 6.

A példán a robot kezdetben vízszintesen a középső mezőn áll, mert így tudja elkapni a balról jobbra első beérkező meteort (amiből itt csak ez az egy van). A 3. időszeletig nem mozdul, mert a meteor mindig ugyanoda, az aktuális pozíciójára érkezik. A 4-5. időszeletben szintén marad, mivel nem lát meteort. A 6. időszeletben kezdi meg a mozgását, ám így a legbaloldalibb mezőbe már nem jut el időben, de balra lép egyet, mert abban az irányban detektál több meteort. Ezután a 7. időszeletben két meteor jön együtt, itt egyet tudna elkapni, mégpedig, ha nem mozdul, ezért ezt teszi. A 8. időszeletben jobbra mozdul, hogy elkapjon egy meteort, majd a következőben szintén jobbra, ám itt nem éri el. A többi meteort is ezeknek megfelelően igyekszik elkapni.

A feladatod tehát meghatározni, hogy hány rakétára lesz szükség, azaz hány meteor jut át a robot védelmi vonalán. A bemeneti fájl első sora két egész számot tartalmaz szóközzel elválasztva, ez a bejövő mátrix magassága és szélessége. Az ez után következő sorok már 0-kat és 1-eseket tartalmaznak az oszlopok számának megfelelő mennyiségben, szóközökkel elválasztva. Ahol 1-est találunk ott van bejövő meteor, ahol pedig 0-t, ott nincs. A legfelső sor jelzi a legközelebbi meteorokat. Ha nincs meteor az első sávban, akkor a robot a legbaloldalibb helyen kezd. A kimenet egyetlen, sortöréssel lezárt sorában az eredmény, azaz a szükséges rakéták száma található.