Üzenet

Aisha és Basma két barát. Aishának van egy M üzenete, amely S bitből álló sorozat (azaz nullák vagy egyesek), amit szeretne elküldeni Basmának. Aisha úgy kommunikál Basmával, hogy **csomagokat** küld neki. Egy csomag 31 bitből álló sorozat, 0 és 30 között sorszámozva. Aisha az M üzenetet szeretné elküldeni Basmának, néhány csomag elküldésével.

Sajnos Kleopátra megzavarta az Aisha és Basma közötti kommunikációt és képes **beszennyezni** a csomagokat. Vagyis minden csomagban Kleopátra pontosan 15 helyen módosíthatja a biteket. Pontosabban, adott egy C sorozat, amelynek hossza 31 és amelyben minden elem vagy 0 vagy 1, a következő jelentéssel:

- $C[i] = 1$ azt jelenti, hogy az i. bitet Kleopátra megváltoztathatja. Ezeket a sorszámú biteket Kleopátra által **módosíthatónak** nevezzük.
- $C[i] = 0$ azt jelenti, hogy a i. bitet Kleopátra nem tudja módosítani.

A C sorozat pontosan 15 darab egyeset és 16 darab nullát tartalmaz. Üzenetküldés közben a Kleopátra által módosítható sorszámú bitek minden csomagra ugyanazok maradnak. Aisha pontosan tudja, hogy Kleopátra melyik 15 darab bitet módosíthatja. Basma csak azt tudja, hogy 15 bitet Kleopátra módosíthat, de nem tudja, melyikeket.

Legyen A egy csomag, amelyet Aisha elküld (ezt **eredeti csomagnak** nevezünk). Legyen B a Basma által fogadott csomag (amit mi **szennyezett csomagnak** nevezünk). Minden i esetén ($0 \leq i < 31$):

- ha Kleopátra nem módosíthatja a i. sorszámú bitet (azaz $C[i] = 0$), akkor Basma az i. bitet úgy kapja meg, ahogyan Aisha küldte (azaz $B[i] = A[i]$),
- egyébként, ha Kleopátra módosíthatja az i. sorszámú bitet (azaz $C[i] = 1$), akkor $B[i]$ értékét Kleopátra dönti el.

Minden egyes csomag elküldése után Aisha azonnal megtudja, mi a megfelelő szennyezett csomag.

Miután Aisha elküldte az összes csomagot, Basma **a küldés sorrendjében** megkapja az összes szennyezett csomagot és rekonstruálnia kell az eredeti M üzenetet.

A feladatod egy stratégia kidolgozása és végrehajtása, ami lehetővé teszi, hogyha Aisha az M üzenetet küldi Basmának, akkor Basma vissza tudja állítani az M üzenetet a szennyezett csomagokból. Konkrétan két eljárást kell implementálnod. Az első eljárás Aisha cselekedeteit hajtja

végre. Adott az M üzenet és a C sorozat, és küldenie kell néhány csomagot, amivel továbbítja az üzenetet Basmának. A második eljárás Basma műveleteit hajtja végre. Megkapja a szennyezett csomagokat és vissza kell állítania az eredeti M üzenetet.

Megvalósítás

Az első eljárás, amelyet implementálnod kell:

```
void send_message(std::vector<bool> M, std::vector<bool> C)
```

- M : S hosszúságú sorozat, az üzenet, amit Aisha el akar küldeni Basmának.
- C : 31 hosszúságú sorozat, a Kleopátra által módosítható biteket adja meg.
- Ez az eljárás **legfeljebb 2100-szor** hívható meg az egyes tesztesetekben.

Ennek az eljárásnak a következő eljárást kell meghívnia egy csomag küldéséhez:

```
std::vector<bool> send_packet(std::vector<bool> A)
```

- A : egy eredeti csomag (31 hosszúságú sorozat) az Aisha által küldött csomag bitjei.
- Ez az eljárás egy szennyezett B csomagot ad vissza, ami a Basma által fogadott biteket adja.
- Ez az eljárás legfeljebb 100 alkalommal hívható meg a `send_message` minden egyes meghívásakor.

A második eljárás, amelyet implementálnod kell:

```
std::vector<bool> receive_message(std::vector<std::vector<bool>> R)
```

- R : Az Aisha által egy `send_message` hívásban küldött csomagokhoz tartozó szennyezett csomagok sorozata, **a küldésük sorrendjében**. R minden eleme egy 31 hosszúságú sorozat, amely egy-egy szennyezett csomag.
- Ennek az eljárásnak egy S bitből álló sorozatot kell visszaadnia, ami megegyezik az eredeti M üzenettel.
- Ez az eljárás minden tesztesetben **többször** hívható, **pontosan egyszer** minden megfelelő `send_message` hívásnál. A `receive_message` **eljárás hívásainak sorrendje** nem feltétlenül egyezik meg a megfelelő `send_message` hívások sorrendjével.

Vedd figyelembe, hogy az értékelőrendszerben a `send_message` és `receive_message` eljárások **két külön programban** vannak meghívva.

Korlátok

- $1 \leq S \leq 1024$
- C pontosan 31 elemet tartalmaz, amelyből 16 egyenlő 0-val és 15 egyenlő 1-gyel.

Részfeladatok és pontozás

Ha bármelyik tesztesetben, a `send_packet` eljárás hívásai nem felelnek meg a fent említett szabályoknak, vagy a `receive_message` eljárás bármely hívásának visszatérési értéke helytelen, az adott teszteset megoldásának pontszáma 0 lesz.

Ellenkező esetben legyen Q a `send_packet` eljárás hívásainak maximális száma a `send_message` összes meghívása között (az összes tesztesetben). Legyen X értéke:

- 1, ha $Q \leq 66$
- 0.95^{Q-66}, ha $66 < Q \leq 100$

Ezután a pontszámot a következőképpen számítják ki:

Részfeladat	Pontszám	További megszorítások
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Nincsenek további megszorítások.

Vedd figyelembe, hogy bizonyos esetekben az értékelő viselkedése lehet **adaptív**. Ez azt jelenti, hogy a `send_packet` által visszaadott értékek nem csak a bemeneti paraméterektől függhetnek, hanem sok mástól is, beleértve az eljárás korábbi hívásainak bemeneti és visszatérési értékeit és az értékelő által generált pszeudo-véletlen számokat. Az értékelő **determinisztikus** abban az értelemben, hogy ha kétszer futtatod és mindkét futásban ugyanazokat a csomagokat küldi el, akkor ugyanazokat a változtatásokat hajtja végre rajtuk.

Példa

Tekintsük a következő függvényhívást:

```
send_message([0, 1, 1, 0],
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Az üzenet, amit Aisha megpróbál Basmának küldeni: $[0, 1, 1, 0]$. A 0 és 15 közötti sorszámú biteket Kleopátra nem tudja megváltoztatni, míg a 16 és 30 közti sorszámú biteket Kleopátra módosíthatja.

A példa kedvéért tegyük fel, hogy Kleopátra az általa módosítható, egymást követő biteket a 0 és az 1 váltakozásával tölti ki, azaz 0-t ír az első általa irányított sorszámú bitre (esetünkben ez a 16.), 1-t ír a második általa irányított sorszámú bitre (ez a 17.), 0-t ír a harmadik általa irányított sorszámú bitre (ez a 18.), és így tovább.

Aisha dönthet úgy, hogy egy csomagban két bitet küld az eredeti üzenetből, a következők szerint: elküldi az első bitet az első 8 hely mindegyikén és a második bitet a következő 8 helyen (ezeket ő állíthatja és Kleopátra nem módosítja).

Aisha így úgy dönt, hogy elküldi a következő csomagot:

```
send_packet([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Vedd figyelembe, hogy Kleopátra módosíthatja a biteket az utolsó 15 helyen, így Aisha tetszőlegesen állíthatja őket, mivel Kleopátra felülírhatja őket.

Kleopátra feltételezett stratégiájával az eljárás a következőket adja vissza: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha úgy dönt, hogy elküldi az M utolsó két bitjét a második csomagban hasonló módon, mint korábban:

```
send_packet([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Kleopátra feltételezett stratégiájával az eljárás a következőket adja vissza: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha küldhet több csomagot, de úgy dönt, hogy mégsem.

Az értékelő ezután a következő eljárást hívja:

```
receive_message([[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
 [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
```

Basma a következőképpen állítja helyre az M üzenetet. Minden csomagból kiveszi az első bitet, amely egymás után kétszer fordul elő, és az utolsó bitet, amely egymás után kétszer fordul elő. Vagyis az első csomagból $[0,1]$ bitet vesz el, a másodikból pedig az $[1,0]$ biteket. Ha összerakja őket, visszaállítja a $[0,1,1,0]$ üzenetet, amely `receive_message` hívás helyes visszatérési értéke.

Megmutatható, hogy Kleopátra feltételezett stratégiájával és 4 hosszúságú üzenetekkel Basma ezen megközelítése helyesen visszaállítja az M üzenetet, függetlenül a C értékétől. Ez azonban általános esetben nem lesz helyes.

Mintaértékelő

A mintaértékelő nem adaptív. Ehelyett Kleopátra az általa vezérelt egymást követő biteket váltakozó 0 és 1 bitekkel tölti ki, a fenti példában leírtak szerint.

Beviteli formátum: **A bemenet első sora egy T egész számot tartalmaz, az esetek számának megadásával.** T eset leírása következik. Mindegyik a következő formátumú:

```
S
M[0] M[1] ... M[S-1]
C[0] C[1] ... C[30]
```

Kimeneti formátum: A mintaértékelő egyesével kiírja az összes T eset eredményét ugyanabban a sorrendben, ahogyan a bemenetben szerepelnek, a következő formátumban:

```
K L
D[0] D[1] ... D[L-1]
```

Itt K a `send_packet` hívások száma, L a hossza, D a `receive_message` által visszaadott üzenetek.