Message

Lamija i Balša su prijatelji koji međusobno komuniciraju na neobičan način. Lamija ima poruku M, koja je zapravo niz od S bitova (niz nula i jedinica), koju želi poslati Balši. Lamija komunicira s Balšaom šaljeći mu **pakete**. Paket je niz od 31-og bita indeksiran od 0 do 30. Lamija želi iskomunicirati poruku M Balšau šaljeći mu pakete.

Nažalost, Lamija u školi ima ljubomorne prijateljice koje žele ukaljati komunikaciju između Lamije i Balše. One metodama špijunaže, lukavstva i prepredenosti mogu promjeniti određenih 15 pozicija u paketu. Formalnije, postoji niz C dužine 31 u kojem je svaki element 0 ili 1, koji opisuje vragolije koje prijateljice izvode na sljedeći način:

- $C[i] = 1$ znači da je poziciju i u paketu **kontrolišu** prijateljice, pa ju mogu ili ne moraju promjeniti po želji.
- $C[i] = 0$ znači da pozicija i nije pod uticajem prijateljica pa ostaje onako kako je Lamija poslala.

Niz C sadrži **tačno** 15 jedinica i 16 nula. Tokom slanja jedne poruke M (kroz više paketa), pozicije koje kontrolišu prijateljice ostaju iste kroz sve pakete. Lamija zna tačno kojih 15 pozicija su riskantne, dok Balša samo zna da je takvih pozicija 15, ali ne zna koje su.

Neka je A paket koji Lamija odluči poslati (još zvan i **originalni paket**). Neka je B paket koji prima Balša (još zvan i **pobrkani paket**). Za svaki i, takav da $0 \leq i < 31$:

- ako prijateljice ne kontrolišu poziciju i ($C[i] = 0$), Balša prima bit i onako kako ga je Lamija poslala ($B[i] = A[i]$),
- inače, ako prijateljice kontrolirajšu poziciju i ($C[i] = 1$), vrijednosti $B[i]$ odlučuju one.

Prijateljice se vole naslađivati pa će odmah nakon što pošalje paket, Lamiji javiti kako izgleda pobrkani paket će načiniti uz poruku:

> Prijatno Lamija!

Nakon što Lamija pošalje sve pakete, Balša odjednom dobija sve pobrkane pakete **u istom onom poretku kako ih je Lamija i slala**. On tada pokušava rekonstruisati poruku M.

Vaš je zadatak dizajnirati i implementirati strategiju koja bi dozvolila da Lamija paketima pošalje poruku M, tako da Balša može rekonstruisati poruku M iz pobrkanih paketa. Točnije, trebate implementirati dvije procedure. Prva procedura izvodi radnje koje čini Lamija. Procedura dobija poruku M i niz C te treba poslati neke pakete kako bi prenjela poruku Balši. Druga procedura

treba imitirati Balšin proces razmišljanja kojim će nakon šoka i nevjerice pokušati iz pobrkanih paketa koje dobije rekonstruirsati poruku M.

Detalji implementacije

Prva procedura glasi ovako:

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

- M: bitovni niz dužine S koji opisuje poruku koju Lamija pokušava slati Balši.
- C: bitovni niz dužine 31 označava pozicije koje kontrolišu prijateljice.
- Ova procedura će biti pozvana **najviše 2100 puta** u svakom test slučaju.

Ta procedura treba pozivati sljedeću za slanje paketa:

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

- A: originalni paket (bitovni niz dužine 31).
- Ova procedura vraća pobrkani paket B.
- Ova procedura smije biti pozvana najviše 100 puta tokom jednog poziva `send_message`.

Druga procedura za implementaciju je:

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

- R: niz koji opisuje pobrkane pakete. Paketi su porijeklom nastali od paketa koje je Lamija poslala tokom nekog poziva `send_message` procedure **i u tačnom poretku**. Svaki član niza R je bitovni niz duljine 31, predstavlja pobrkani paket.
- Ova procedura treba vraćati bitovni niz dužine S koji bi trebao biti jednak originalnoj poruci M.
- Ova procedura će biti pozvana **više puta** u svakom test podatku, **tačno jednom** za svaki `send_message` poziv. **Poredak** `receive_message` poziva nije nužno isti kao onaj `send_message` poziva.

Napomenimo da će `send_message` i `receive_message` procedure biti pozvane u dva različita programa tokom gradinga..

Ograničenja

- $1 \leq S \leq 1024$
- C ima tačno 31 elemenata, od kojih su 16 jednaki 0, a 15 su jednaki 1.

Podzadaci i Grejding

Ako u nekom test podatku, pozivi procedure `send_packet` nijesu u skladu s pravilima napomenutima gore, ili je izlaz procedure `receive_message` kriv, bodovi ostvareni na tom test podatku iznosiće 0.

Inače, neka je Q maksimalni broj poziva procedure `send_packet` među svim pozivima `send_message` među svim test podacima. Neka je nadalje X jednak:

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

Tada je broj bodova ostvaren po sljedećem kriteriju:

Podzadatak	Poeni / Bodovi	Dodatna ograničenja
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Nema dodatnih ograničenja.

Napomena: U nekim slučajevima ponašanje GREJDERA može biti **adaptivno**. To znači da vrijednosti koje vraća 'send_packet' mogu zavisiti ne samo od njenih ulaznih argumenata, već i od mnogih drugih stvari, uključujući ulaze i povratne vrijednosti prethodnih poziva ovoj proceduri i pseudo-nasumične brojeve koje generiše grejder. Grejder je **deterministički** u smislu da, ako ga pokrenete dva puta i u oba slučaja pošaljete iste pakete, napraviće iste promjene na njima.

Primjer

Ajde da promotrimo sljedeći primjer.

```
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])
```

Poruka koju Lamija pokušava da pošalje Balši jeste $[0, 1, 1, 0]$. Bitovi na pozicija od 0 do 15 ne mogu da budu promjenjeni od strane prijateljica, dok oni na pozicijama od 16 do 30 to mogu da budu.

Radi jednostavnosti, ajmo da pretpostavimo da će prijateljice popunjavati pozicije koje kontrolišu alternirajući sa 0 i 1, tj. staviće 0 na prvu poziciju koju kontrolišu (pozicija 16 u ovom slučaju), 1 na drugu (pozicija 17), 0 na treću koju kontrolišu (pozicija 18), i tako dalje.

Lamija će svakim paketom slati 2 bita iz originalne poruke na sljedeći način: prvi od njih zapisat će na svih prvih 8 pozicija koje kontroliše te drugi na preostalih 8.

Dakle ovako:

```
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])
```

Primjetimo da pošto prijateljice nasumice mogu promjeniti zadnjih 15 pozicija, nije bitno što Lamija tamo pošalje kako one mogu biti prebrisane. Sa pretpostavljenom strategijom pobrkani paket izgleda ovako: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Lamije sada odlučuju poslati posljednja dva bita poruke M u drugom paketu na sličan način:

```
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])
```

Pobrkani paket sada glasi: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Lamija može da šalje još, ali ne želi.

Grader sada izvrši sljedeći poziv:

```
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]])
```

Balša rekonstruiše poruku M sa sljedećom strategijom. Od svakog paketa uzme prvi bit koji se pojavljuje dva puta za redom i zadnji takav. Odnosno, od prvog pobrkanog paketa dobija $[0,1]$, a iz drugog $[1,0]$. Spajajući dobija poruku $[0,1,1,0]$, Što je i točan izlaz za proceduru `receive_message`.

Može se pokazati da uz pretpostavljenu strategiju prijateljica i za poruke dužine 4, ovaj pristup Basme ispravno obnavlja M, bez obzira na vrijednost C. Međutim, nije tačan u opštem slučaju.

Grejder

Priloženi grejder nije adaptivan. Umjesto toga, prijateljice popunjavaju uzastopne bitove koje kontrolišu naizmjenično sa 0 i 1 bitovima, kao što je opisano u prethodnom primjeru.

Ulazni format: **Prva linija sadrži broj T, broj scenarija.** T scenarija sljedi. Svaki od scenarija izgleda ovako:

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

Izlazni format: Pirloženi grader zapisuje svaki od T scenarija u poretku u kojem su navedeni u ulazu ovako:

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

Ovdje je, K broj poziva `send_packet`, D je poruka koju je vratio poziv od `receive_message` i L je njena dužina.