# Message

Lamija i Vedran 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 Vedranu. Lamija komunicira s Vedranom šaljeći mu **pakete**. Paket je niz od $31$-og bita indeksiran od $0$ do $30$. Lamija želi iskomunicirati poruku $M$ Vedranu šaljeći mu pakete.

Nažalost, Lamija u školi ima ljubomorne prijateljice koje žele ukaljati komunikaciju između Lamije i Vedrana. One metodama špijunaže, lukavstva i prepredenosti mogu promjeniti određenih 15 pozicija u paketu. Formalnije, postoji niz $C$ duljine $31$ 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 **kontroliraju** prijateljice, te ju mogu ili ne moraju promjeniti po želji.
- $C[i] = 0$ znači da pozicija $i$ nije pod utjecajem prijateljica te ostaje onako kako je Lamija poslala.

Niz $C$ sadrži **točno** $15$ jedinica i $16$ nula. Tijekom slanja jedne poruke $M$ (kroz više paketa), pozicije koje kontroliraju prijateljice ostaju iste kroz sve pakete. Lamija zna točno kojih $15$ pozicija su riskantne, dok Vedran 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 Vedran (još zvan i **pobrkani paket**). Za svaki $i$, takav da $0 \leq i < 31$:

- ako prijateljice ne kontroliraju poziciju $i$ ($C[i] = 0$), Vedran prima bit $i$ onako kako ga je Lamija poslala ($B[i] = A[i]$),
- inače, ako prijateljice kontroliraju 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 koji će načiniti uz poruku:

> Prijatno Lamija!

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

Vaš je zadatak dizajnirati i implementirati strategiju koja bi dozvolila da Lamija paketima pošalje poruku $M$, tako da Vedran može rekonstruirati 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 Vedranu. Druga procedura

treba imitirati Vedranov proces razmišljanja kojim će nakon šoka i nevjerice pokušati iz pobrkanih paketa koje dobije rekonstruirati poruku $M$.

## Implementation Details

Prva procedura glasi ovako:

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

- $M$: bitovni niz duljine $S$ koji opisuje poruku koju Lamija pokušava slati Vedranu.
- $C$: bitovni niz duljine $31$ označava pozicije koje kontroliraju prijateljice.
- Ova procedura će biti pozvana **najviše 2100 puta** u svakom test podatku.

Ta procedura treba pozivati sljedeću za slanje paketa:

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

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

Druga procedura za implementaciju je:

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

- $R$: niz koji sadrži pobrkane pakete. Paketi su porijeklom nastali od paketa koje je Lamija poslala tijekom nekog poziva `send_message` procedure **i u točnom poretku**. Svaki član niza $R$ je bitovni niz duljine $31$, predstavlja pobrkani paket.
- Ova procedura treba vraćati bitovni niz duljine $S$ koji bi trebao biti jednak originalnooj poruci $M$.
- Ova procedura će biti pozvana **više puta** u svakom test podatku, **točno 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 različitim programima tijekom gradinga..

## Constraints

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

## Subtasks and Scoring

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

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 $66 < Q \leq 100$
- $0$, if $100 < Q$

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. |

Napomenimo da se grader u nekim slučajevima ponaša **adaptivno**. To znači da pobrkani paketi koje šalje `send_packet` mogu ovisiti o ulaznim argumentima kao i o prijašnjim pobrkanim paketima prethodnih poziva `send_packet`.

### Example

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, 1])
```

Poruka koju Lamija pokušava da pošalje Vedranu jest $[0, 1, 1, 0]$. 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 se prijateljice ponašaju deterministički, one će popunjavati pozicije koje kontroliraju alternirajući sa $0$ i $1$, tj. stavit će $0$ na prvu poziciju koju kontroliraju (pozicija $16$ u ovom slučaju), $1$ na drugu (pozicija $17$), $0$ na treću koju kontroliraju (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 prvij $8$ pozicija koje kontrolira 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: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

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: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

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]])
```

Vedran rekonstruktuje 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`.

It can be shown that with the assumed strategy of Cleopatra and for messages of length $4$, this approach of Basma correctly recovers $M$, regardless of the value of $C$. However, it is not correct in the general case.

## Priloženi Grader

Priloženi grader nije adaptivan kao onaj pravi. Umjesto toga, prijateljice se ponašaju deterministički, uvijek će popunjavati svoje pozicije alternirajući s $0$ i $1$, kako je opisano gore (makar one ne moraju biti uzastopne).

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 duljina.