# Ziņojums

Aiša un Basma ir divas draudzenes, kas sarakstās viena ar otru. Aišai ir ziņojums $M$, kas ir $S$ bitu (tas ir, vieninieku un nuļļu) virkne, ko viņa vēlas aizsūtīt Basmai. Aiša komunicē ar Basmu sūtot viņai ziņojuma **paketes**. Pakete ir virkne no $31$ bita, kas numurēti no $0$ līdz $30$. Aiša vēlas aizsūtīt ziņojumu $M$ Basmai, nosūtot viņai kādu skaitu pakešu.

Diemžēl Kleopatra var noklausīties komunikācijas kanālu starp Aišu un Basmu un var **bojāt** tajā sūtītās paketes. Tas nozīmē, ka katrā paketē Kleopatra var nomainīt bitus tieši $15$ pozīcijās.

Konkrēti, ir masīvs $C$, kura garums ir $31$, kurā katrs elements ir $0$ vai $1$ ar šādu nozīmi:

- $C[i] = 1$ norāda, ka $i$-to bitu Kleopatra var nomainīt. Mēs šos bitus saucam par Kleopatras **kontrolētiem**.
- $C[i] = 0$ norāda, ka Kleopatra nevar mainīt $i$-to bitu.

Masīvs $C$ satur precīzi $15$ vieniniekus un $16$ nulles. Aizsūtot ziņojumu, Kleopatras kontrolētā bitu kopa paliek nemainīga visām paketēm. Aiša precīzi zina, kurus $15$ bitus kontrolē Kleopatra. Basma zina tikai to, ka $15$ bitus kontrolē Kleopatra, bet viņa nezina, kurus tieši.

Pieņemsim, ka $A$ ir pakete, kuru Aiša nolemj nosūtīt (ko mēs saucam par **oriģinālo paketi**). Lai $B$ ir pakete, ko saņem Basma (ko mēs saucam par **bojātām paketēm**). Katram $i$, kur $0 \leq i < 31$:

- ja Kleopatra nekontrolē $i$-to bitu ($C[i] = 0$), Basma saņem $i$-to bitu, ko nosūtīja Aiša ($B[i] = A[i]$),
- pretējā gadījumā, ja Kleopatra kontrolē $i$-to bitu ($C[i] = 1$), $B[i]$ vērtību izlemj Kleopatra.

Uzreiz pēc katras paketes nosūtīšanas, Aiša uzzina, kāda ir attiecīgā bojātā pakete.

Kad Aiša ir nosūtījusi visas paketes, Basma saņem visas bojātās paketes **to izsūtīšanas secībā**, un viņai ir jārekonstruē sākotnējais ziņojums $M$.

Tavs uzdevums ir izplānot un implementēt stratēģiju kas ļautu Aišai aizsūtīt ziņojumu $M$ Basmai, lai Basma varētu atjaunot ziņojumu $M$ no bojātajām paketēm. Konkrēti, jums vajadzētu īstenot divas procedūras. Pirmā procedūra veic Aišas darbības. Procedūrai tiek dots ziņojums $M$ un masīvs $C$, un tai jānosūta paketes, lai aizsūtītu ziņojumu Basmai. Otrā procedūra veic Basmas darbības. Procedūrai tiek dotas bojātās paketes, tai ir jāatjauno sākotnējais ziņojums $M$.

## Implementēšanas detaļas

Pirmā procedūra, kas jums jāimplementē, ir:

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

- $M$: masīvs garumā $S$, kas apraksta ziņojumu, ko Aiša vēlas aizsūtīt Basmai.
- $C$: masīvs garumā $31$, kas norāda bitu pozīcijas, ko kontrolē Kleopatra.
- Šo procedūru katrā testā var izsaukt **ne vairāk kā 2100 reizes**.

Šai procedūrai ir jāizsauc sekojoša procedūra, lai nosūtītu paketi:

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

- $A$: oriģinālā pakete (masīvs garumā $31$), kas reprezentē Aišas nosūtītos bitus.
- Šī procedūra atgriež bojāto paketi $B$, kas reprezentē bitus, ko saņems Basma.
- Šo procedūru var izsaukt ne vairāk kā $100$ reizes katrā `send_message` izpildes reizē.

Otrā procedūra, kas jums jāimplementē, ir:

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

- $R$: masīvs, kas apraksta bojātās paketes. Paketes ir radušās no oriģinālajām paketēm, ko nosūtīja Aiša vienā `send_message` izsaukuma reizē, un tās ir dotas **secībā, kādā tās nosūtīja** Aiša. Katrs $R$ elements ir masīvs garumā $31$, kas reprezentē bojāto paketi.
- Šai procedūrai jāatgriež masīvs ar $S$ bitiem, kas ir vienāds ar oriģinālo ziņojumu $M$.
- Šī procedūra var tikt izsaukta **vairākas reizes** katrā testā, **tieši vienreiz** katram atbilstošam `send_message` izsaukumam. **Secība** `receive_message` **procedūru izsaukumiem** var nesakrist ar secību atbilstošajiem `send_message` izsaukumiem.

Ņemiet vērā, ka vērtēšanas sistēmā procedūras `send_message` un `receive_message` tiek izsauktas **divās atsevišķās programmās**.

## Ierobežojumi

- $1 \leq S \leq 1024$
- $C$ satur tieši $31$ elementu, no kuriem $16$ ir vienādi ar $0$ un $15$ ir vienādi ar $1$.

## Apakšuzdevumi un vērtēšana

Ja kādā no testiem, procedūras `send_packet` izsaukumi neatbilst iepriekš minētajiem noteikumiem vai jebkuras procedūras `receive_message` izsaukuma atgriešanas vērtība ir nepareiza, jūsu risinājums iegūs $0$ punktus šajā testā.

Pretējā gadījumā, pieņemsim, ka visos testos $Q$ ir maksimālais procedūras `send_packet` izsaukumu skaits starp visiem `send_message` izsaukumiem. Pieņemsim, ka $X$ ir vienāds ar:

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

Tad iegūtais punktu skaits tiek aprēķināts šādi:

| Apakšuzdevums | Punkti | Papildu ierobežojumi |
|---|---|---|
| 1 | $10 \cdot X$ | $S \leq 64$ |
| 2 | $90 \cdot X$ | Bez papildu ierobežojumiem. |

Ņemiet vērā, ka dažos gadījumos vērtētāja programmas uzvedība var būt **adaptīva**. Tas nozīmē, ka `send_packet` atgrieztās vērtības var būt atkarīgas ne tikai no ievades argumentiem, bet arī no daudzām citām lietām, ieskaitot ievades argumentus un atgriešanas vērtības no iepriekšējiem šīs procedūras izsaukumiem un pseidogadījumskaitļiem, ko uzģenerēja vērtētāja programma. Vērtētāja programma ir **deterministiska** tādā nozīmē, ka, ja to izpilda divas reizes, un abos gadījumos jūs sūtāt vienas un tās pašas paketes, tajās tiks veiktas tādas pašas izmaiņas.

## Piemērs

Aplūkosim šādu izsaukumu:

```
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])
```

Ziņojums, ko Aiša mēģina aizsūtīt Basmai, ir $[0, 1, 1, 0]$. Bitus, kas numurēti no $0$ līdz $15$, Kleopatra nevar mainīt, savukārt bitus, kas numurēti no $16$ līdz $30$, Kleopatra var izmainīt.

Tagad pieņemsim, ka Kleopatra aizpilda secīgos bitus, kurus viņa kontrolē, pārmaiņus ar $0$ un $1$ , tas ir, viņa piešķir $0$ pirmajam bitam, ko viņa kontrolē (mūsu gadījumā $16$-tajam bitam ), $1$ otrajam bitam, ko viņa kontrolē ($17$-tajam bitam), $1$ trešajam bitam, ko viņa kontrolē ($18$-tajam bitam), un tā tālāk.

Aiša var izlemt nosūtīt divus bitus no sākotnējā ziņojuma vienā paketē šādi: viņa nosūtīs pirmo bitu pirmajos $8$ bitos, ko viņa kontrolē un otro bitu sekojošajos $8$ bitos, ko viņa kontrolē.

Tāpēc Aiša nosūta šādu paketi:

```
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])
```

Ņemiet vērā, ka Kleopatra var mainīt pēdējos $15$ bitus, tādēļ Aiša var tos iestatīt patvaļīgi, jo tie var tikt pārrakstīti. Izmantojot pieņemto Kleopatras stratēģiju, procedūra atgriež: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aiša nolemj nosūtīt pēdējos divus $M$ bitus otrajā paketē līdzīgā veidā kā iepriekš:

```
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])
```

Izmantojot pieņemto Kleopatras stratēģiju, procedūra atgriež: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aiša var nosūtīt vairāk pakešu, bet viņa izvēlas to nedarīt.

Pēc tam vērtētāja programma veic sekojošo procedūras izsaukumu:

```
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]])
```

Basmai jāatjauno sākotnējo ziņojumu $M$ šādi. No katras paketes viņa paņem pirmo bitu, kas atkārtojas divas reizes pēc kārtas, un pēdejo bitu, kas atkārtojas divas reizes pēc kārtas. Tas ir, no pirmās paketes viņa paņem bitus $[0,1]$ un no otrās paketes viņa paņem bitus $[1,0]$. Saliekot tos kopā, viņa atjauno ziņojumu $[0,1,1,0]$, kas ir pareizā atgriešanas vērtība šim `receive_message` izsaukumam.

Var pierādīt, ka ar pieņemto Kleopatras stratēģiju un ziņojumiem, kuru garums ir $4$ , šī Basmas pieeja pareizi atgūst $M$ neatkarīgi no $C$ vērtības. Tomēr vispārīgā gadījumā šāda pieeja nav pareiza.

## Paraugvērtētājs

Paraugvērtētāja programma nav adaptīva. Tā vietā Kleopatra aizpilda secīgus bitus, kurus viņa kontrolē, pārmaiņus ar $0$ un $1$, kā aprakstīts iepriekš minētajā piemērā.

Ievaddatu formāts: **Ievaddatu pirmajā rindā ir vesels skaitlis $T$, norādot scenāriju skaitu.** Tālāk seko $T$ scenāriji. Katrs no tiem ir dots šādā formātā:

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

Izvaddatu formāts: Paraugvērtētāja programma izvada rezultātu katram no $T$ scenārijiem tādā pašā secībā, kādā tie ir norādīti ievadā, šādā formātā:

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

Šeit $K$ ir `send_packet` izsaukumu skaits, $D$ ir ziņojums, ko atgriež `receive_message` un $L$ ir tā garums.