Berichten

Aisha en Basma zijn twee vriendinnen die elkaar berichten sturen. Aisha heeft een bericht M, een reeks van S bits (i.e. nullen en enen), dat ze wil verzenden naar Basma. Aisha communiceert met Basma door haar **pakketten** te sturen. Een pakket is een reeks van 31 bits, geïndexeerd van 0 tot en met 30. Aisha wil het bericht M naar Basma sturen door haar een aantal pakketten te sturen.

Helaas heeft Cleopatra de communicatie tussen Aisha en Basma verstoord en zijn kan de pakketten **besmetten**. Dat wil zeggen dat Cleopatra in elk pakket bits op precies 15 indices kan wijzigen. Om precies te zijn is er een array C van lengte 31, waarin elk element óf 0 óf 1 is, met de volgende betekenis:

- $C[i] = 1$ geeft aan dat het bit met index i door Cleopatra kan worden gewijzigd. Wij noemen deze indices **onder controle** van Cleopatra.
- $C[i] = 0$ geeft aan dat Cleopatra de bit met index i niet kan veranderen.

De array C bevat precies 15 enen en 16 nullen. Tijdens het verzenden van het bericht, M, blijft de door Cleopatra beheerde set indices voor alle pakketten hetzelfde. Aisha weet precies welke 15 indices door Cleopatra worden beheerd. Basma weet alleen dat 15 indices worden beheerd door Cleopatra, maar ze weet niet welke indices dat zijn.

We noemen het pakket dat Aisha gaat verzenden A (ook wel het **originele pakket**). We noemen het pakket dat Basma binnen krijgt B (ook wel het **besmette pakket**). Voor elke i, zodat $0 \leq i < 31$:

- als de bit met index i niet onder controle is van Cleopatra ($C[i] = 0$), krijgt Basma de bit binnen die Aisha had verzonden ($B[i] = A[i]$)
- als de bit met index i wel onder controle is van Cleopatra ($C[i] = 1$), dan kan Cleopatra beslissen wat de waarde is van de bit $B[i]$ die Basma binnen krijgt

Gelijk nadat Aisha een pakket verzend, krijgt zij het hele corresponderende besmette pakket te weten.

Jouw taak is om een strategie te bedenken en te implementeren die Aisha helpt om het bericht M te verzenden aan Basma, zodat Basma M kan reconstrueren van de besmette pakketten. Om precies te zijn moet je twee procedures implementeren. De eerste procedure helpt Aisha. De procedure krijgt een bericht M en de array C, en moet een aantal pakketten versturen om het

bericht over te brengen aan Basma. De tweede procedure helpt Basma. De procedure krijgt de besmette pakketten en moet het originele bericht M reconstrueren.

Implementatie Details

De eerste procedure die je moet implementeren is:

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

- M: een array met lengte S, het bericht dat Aisha wil verzenden naar Basma.
- C: een array met lengte 31 die de indices aangeeft van de bits die onder controle van Cleopatra zijn.
- Deze procedure zal **niet meer dan 2100 keer** worden aangeroepen in een test case.

Deze procedure moet de volgende procedure aanroepen om een pakket te verzenden:

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

- A: een origineel pakket (een array van lengte 31) die de bits vertegenwoordigen die Aisha verzend.
- Deze procedure returnt een besmet pakket B dat de bits laat zien die door Basma worden ontvangen.
- Deze procedure kan maximaal 100 keer worden aangeroepen bij elke aanroep van `send_message`.

De tweede procedure die je moet implementeren is:

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

- R : array die de besmette pakketten beschrijft. De pakketten zijn afkomstig van pakketten die door Aisha in één `send_message` oproep zijn verzonden en worden **in de volgorde waarin ze door Aisha zijn verzonden** gegeven. Elk element van R is een array met lengte 31 en beschrijft een besmet pakket.
- Deze procedure moet een array van S bits returnen. die gelijk is aan het oorspronkelijke bericht M.
- Deze procedure kan **meerdere keren** in elke testcase worden aangeroepen, **exact één keer** voor elke overeenkomstige `send_message` aanroep. De **volgorde** van `receive_message` **procedure-aanroepen** is niet noodzakelijkerwijs hetzelfde als de volgorde van de overeenkomstige `send_message` aanroepen.

Houd er rekening mee dat in het beoordelingssysteem de procedures `send_message` en `receive_message` in **twee afzonderlijke programma's** worden aangeroepen.

Beperkingen

- $1 \leq S \leq 1024$
- C heeft exact 31 elementen, waarvan 16 gelijk zijn aan 0 en 15 gelijk zijn aan 1.

Subtaken en Deelscores

Als in een van de testgevallen, de aanroepen van de procedure `send_packet` niet voldoen aan de hierboven genoemde regels, of de returnwaarde van een van de aanroepen van de procedure `receive_message` onjuist is, dan krijgt jouw oplossing voor die testcase 0 punten.

Noem Q het maximale aantal aanroepen van de procedure `send_packet` over alle aanroepen van `send_message` in alle testcases. Definieer X als:

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

Dan kan de score worden berekend met:

Subtaak	Score	Extra Beperkingen
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Geen extra beperkingen.

Houd er rekening mee dat het gedrag van de beoordelaar in sommige gevallen **adaptief** is. Dit betekent dat de waardes gereturnt door `send_packet` niet alleen afhankelijk zijn van de input argumenten, maar ook van veel andere dingen, zoals input en output waardes van vorige calls naar deze procedure and pseudo-random getallen gegenereerd door de grader. De grader is **deterministisch**. Dit betekent dat als in twee runs van de grader exact dezelfde pakketten worden verstuurd door Aisha, dezelfde veranderingen worden gedaan door Cleopatra.

Voorbeeld

Bekijk de volgende call.

```
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])
```

Het bericht dat Aisha probeert te verzenden aan Basma is $[0, 1, 1, 0]$. De bits met indices van 0 tot 15 kunnen niet door Cleopatra worden gewijzigd, terwijl de bits met indices van 16 tot en met 30 door Cleopatra veranderd kunnen worden.

Voor dit voorbeeld: laten we aannemen dat Cleopatra opeenvolgende bits die ze controleert met afwisselend 0 en 1 vult, dus ze zet de bits op de volgende manier: 0 voor de eerste index die zij controleert (index 16 in ons geval), 1 voor de tweede index die zij controleert (index 17), 0 voor de derde index die zij controleert (index 18), enzovoort.

Aisha kan ervoor kiezen om twee bits uit het originele bericht in één pakket te versturen, en wel als volgt: Ze zal het eerste bit verzenden naar de eerste 8 indices die ze controleert en het tweede stukje bij de volgende 8 indices die zij controleert.

Aisha besluit vervolgens het volgende pakket te versturen:

```
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])
```

Houd er rekening mee dat Cleopatra bits kan veranderen met de laatste 15 indices, dus Aisha zou ze willekeurig kunnen instellen, omdat ze toch overschreven kunnen worden. Met de veronderstelde strategie van Cleopatra komt de procedure terug: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha besluit de laatste twee bits van M in het tweede pakket te versturen op een soortgelijke manier als voorheen:

```
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])
```

Met de veronderstelde strategie van Cleopatra returnt de procedure $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha kan meer pakketten sturen, maar ze kiest ervoor dat niet te doen.

De grader roept vervolgens de volgende procedure aan:

```
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 herstelt bericht M als volgt. Van elk pakket neemt ze het eerste bit dat twee keer achter elkaar voorkomt, en het laatste bit dat twee keer achter elkaar voorkomt. Dat wil zeggen dat ze uit het eerste pakket bits $[0,1]$ neemt, en uit het tweede pakket neemt ze bits $[1,0]$. Door ze samen te voegen, herstelt ze het bericht $[0,1,1,0]$, de juiste returnwaarde voor deze aanroep van `receive_message`.

Er kan worden aangetoond dat met de veronderstelde strategie van Cleopatra en voor berichten van lengte 4, deze manier van herstellen van Basma altijd M correct hersteld, ongeacht de waarde van C. In het algemeen is dit echter niet juist.

Voorbeeldgrader

De voorbeeldgrader is niet adaptief. In plaats daarvan vult Cleopatra de bits die zij onder controle heeft met alternerend 0 en 1.

Invoerformaat:**De eerste regel van de invoer bevat een integer T, het aantal scenario's.** T scenario's volgens. Elk scenario wordt gegeven in het volgende formaat:

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

Uitvoerformaat: De voorbeeldgrader schrijft het resultaat van elk van de T scenario's in dezelfde volgorde als in de invoer in het volgende formaat:

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

Hier is K het aantal calls van `send_packet`, D is het bericht gereturnt door `receive_message` and L is de lengte van dit bericht.