Порака

Софија и Фара се две пријателки кои се допишуваат една со друга. Софија има порака M, која е низа од S битови (т.е. нули или единици), што би сакала да ѝ ги испрати на Фара. Софија комуницира со Фара така што ѝ праќа **пакети**. Пакет е низа од 31 бита индексирани од 0 до 30 . Софија би сакала да ја испрати пораката M до Фара испраќајќи ѝ одреден број пакети.

За жал, Клеопатра ја компромитирала комуникацијата помеѓу Софија и Фара и е во состојба да ги **извалка** пакетите. Односно, во секој пакет Клеопатра може да менува битови на точно 15 индекси (позиции). Конкретно, постои низа C со должина 31, во која секој елемент е или 0 или 1, со следново значење:

- $C[i] = 1$ означува дека битот со индекс i може да биде променет од Клеопатра. Овие индекси ги нарекуваме **контролирани** од Клеопатра.
- $C[i] = 0$ означува дека битот со индекс i не може да биде променет од Клеопатра.

Низата C содржи точно 15 единици и 16 нули. Додека се испраќа пораката M, множеството индекси контролирани од Клеопатра останува исто за сите пакети. Софија точно знае кои 15 индекси се контролирани од Клеопатра. Фара знае само дека 15 индекси се контролирани од Клеопатра, но таа не знае кои индекси.

Нека A е пакет што Софија одлучува да го испрати (кој го нарекуваме **оригинален пакет**). Нека B е пакетот што го прима Фара (кој го нарекуваме **извалкан пакет**). За секое i, така што $0 \leq i < 31$:

- ако Клеопатра не го контролира битот со индекс i ($C[i] = 0$), Фара го добива битот i онака како што е испратен од Софија ($B[i] = A[i]$),
- во спротивно, ако Клеопатра го контролира битот со индекс i ($C[i] = 1$), вредноста на $B[i]$ ја одлучува Клеопатра.

Веднаш по испраќањето на секој пакет, Софија дознава кој е соодветниот извалкан пакет.

Откако Софија ќе ги испрати сите пакети, Фара ги прима сите извалкани пакети **по редоследот по којшто биле испратени** и мора да ја реконструира оригиналната порака M .

Ваша задача е да смислите и имплементирате стратегија што ќе и овозможи на Софија да ја испрати пораката M до Фара, така што Фара ќе може да ја поврати M од извалканите

пакети. Поточно, треба да имплементирате две процедури. Првата процедура ги извршува акциите на Софија. На неа се задава порака M и низата C , и треба да испрати некои пакети за да ја пренесе пораката до Фара. Втората процедура ги извршува акциите на Фара. На неа се задаваат извалканите пакети и треба да ја врати оригиналната порака M.

Имплементациски детали

Првата процедура што треба да ја имплементирате е:

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

- M: низа со должина S што ја опишува пораката што Софија сака да и ја испрати на Фара.
- C: низа со должина 31 што ги означува индексите на битови контролирани од Клеопатра.
- Оваа процедура може да се повика **најмногу 2100 пати** во секој тест случај.

Оваа процедура треба да ја повика следната процедура за да испрати пакет:

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

- A: оригинален пакет (низа со должина 31) што ги претставува битовите испратени од Софија.
- Оваа процедура враќа извалкан пакет B што ги претставува битовите што ќе ги прими Фара.
- Оваа процедура може да се повика најмногу 100 пати во секое повикување на `send_message` .

Втората процедура што треба да ја имплементирате е:

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

- R: низа што ги опишува извалканите пакети. Пакетите потекнуваат од пакети испратени од Софија во еден повик до `send_message` и се дадени **по редоследот по кој биле испратени** од Софија. Секој елемент од R е низа со должина 31, што претставува извалкан пакет.
- Оваа процедура треба да врати низа од S битови што е еднаква на оригиналната порака M .
- Оваа процедура може да се повика **повеќе пати** во секој тест случај, **точно еднаш** за секој соодветен повик до `send_message` . **Редоследот на повици до процедурата** `receive_message` не мора да биде ист со редоследот на соодветните повици до `send_message`.

Да забележиме дека во оценувачкиот систем процедурите `send_message` и `receive_message` се повикуваат во **две одделни програми**.

Ограничувања

- $1 \leq S \leq 1024$
- C има точно 31 елементи, од кои 16 се еднакви на 0 и 15 се еднакви на 1.

Подзадачи и бодување

Ако во кој било од тест случаите, повиците до процедурата `send_packet` не ги почитуваат правилата наведени погоре, или пак ако повратната вредност на кој било од повиците до процедурата `receive_message` не е точна, поените за вашето решение за соодветниот тест случај ќе бидат 0.

Инаку, нека Q е максималниот број на повици до процедурата `send_packet` помеѓу сите повици до `send_message` низ сите тест случаи. Исто така, нека X е еднакво на:

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

Тогаш, поените се пресметуваат на следниот начин:

Подзадача	Поени	Дополнителни ограничувања
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Нема дополнителни ограничувања.

Да забележиме дека во некои случаи однесувањето на оценувачот може да биде **адаптивно**. Ова значи дека вредностите вратени од `send_packet` може да зависат не само од нејзините влезни аргументи, туку и од многу други нешта, вклучувајќи ги влезовите и повратните вредности на претходните повици до оваа процедура, како и псевдо-случајни броеви генерирани од оценувачот. Оценувачот е **детерминистички** во смисла на тоа дека ако го извршите два пати и во двете извршувања ги испратите истите пакети, тој ќе ги направи истите промени врз нив.

Пример

Да го разгледаме следниот повик.

```
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])
```

Пораката што Софија се обидува да ја испрати до Фара е $[0, 1, 1, 0]$. Битовите со индекси од 0 до 15 не може да ги менува Клеопатра, додека битовите со индекси од 16 до 30 може да ги менува Клеопатра.

За потребите на овој пример, да претпоставиме дека Клеопатра ги пополнува последователните битови што ги контролира наизменично со 0 и 1, односно таа доделува 0 на првиот индекс што таа го контролира (индекс 16 во нашиот случај), 1 на вториот индекс што таа го контролира (индекс 17), 0 на третиот индекс што таа го контролира (индекс 18), и така натаму.

Софија може да одлучи да испрати два бита од оригиналната порака во еден пакет на следниов начин: таа ќе го испрати првиот бит на првите 8 индекси што ги контролира и вториот бит на следните 8 индекси што таа ги контролира.

Софија потоа одлучува да го испрати следниот пакет:

```
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])
```

Забележете дека Клеопатра може да менува битови на последните 15 индекси, па Софија може да ги постави произволно, бидејќи може да бидат пребришани. Со претпоставената стратегија на Клеопатра, процедурата враќа: $[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$.

Софија одлучува да ги испрати последните два бита од M во вториот пакет на сличен начин како претходно:

```
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])
```

Со претпоставената стратегија на Клеопатра, процедурата враќа: $[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]$.

Софија може да испраќа уште пакети, но таа одлучува да не го прави тоа.

Оценувачот потоа го прави следниот повик до процедура:

```
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]])
```

Фара ја враќа пораката M на следниов начин. Од секој пакет таа го зема првиот бит што се појавува двапати по ред, и последниот бит што се појавува двапати по ред. Односно, од првиот пакет, таа ги зема битовите $[0, 1]$, а од вториот пакет таа ги зема битовите $[1, 0]$. Со нивно спојување, таа ја враќа пораката $[0, 1, 1, 0]$, што е точната повратна вредност за овој повик до `receive_message` .

Може да се покаже дека со претпоставената стратегија на Клеопатра и за пораки со должина 4, овој пристап на Фара правилно го враќа M, без оглед на вредноста на C. Сепак, тоа не е точно во општ случај.

Пример-оценувач

Пример-оценувачот не е адаптивен. Наместо тоа, Клеопатра ги пополнува последователните битови што ги контролира наизменично со 0 и 1, како што беше објаснето во примерот погоре.

Формат на влез: **Првата линија од влезот содржи еден цел број T, што го специфицира бројот на сценарија.** Следуваат T сценарија. Секое од нив е зададено во следниот формат:

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

Формат на излез: Пример-оценувачот го запишува резултатот од секое од T-те сценарија по истиот редослед како што тие се зададени во влезот, во следниот формат:

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

Овде, K е бројот на повици до `send_packet`, D е пораката вратена од `receive_message`, а L е нејзината должина.