Pesan

Aisha dan Basma berteman satu sama lain. Aisha memiliki sebuah pesan M, yang merupakan sebuah barisan sepanjang S bit (yakni, nol atau satu), yang ingin ia kirimkan ke Basma. Aisha berkomunikasi dengan Basma dengan cara mengirimkan **paket-paket**. Sebuah paket merupakan sebuah barisan sepanjang 31 bit dengan indeks dari 0 hingga 30. Aisha ingin mengirimkan pesan M kepada Basma dengan mengirimkannya beberapa paket.

Sayangnya, Cleopatra menyadap komunikasi antara Aisha dan Basma dan dapat **memanipulasi** paket. Persisnya di setiap paket, Cleopatra dapat mengganti bit-bit pada tepat 15 indeks. Secara spesifik, terdapat sebuah *array* C sepanjang 31 yang seluruh elemennya berupa 0 atau 1, dengan pengertian sebagai berikut:

- $C[i] = 1$ menyatakan bahwa bit pada indeks i dapat diganti oleh Cleopatra. Kita sebut indeks-indeks ini **dikontrol** oleh Cleopatra.
- $C[i] = 0$ menyatakan bahwa bit pada indeks i tidak dapat diganti oleh Cleopatra.

Array C berisi tepat 15 angka satu dan 16 angka nol. Selama mengirimkan pesan, himpunan indeks-indeks yang dikontrol oleh Cleopatra tetaplah sama. Aisha tahu secara persis semua 15 indeks yang dikontrol oleh Cleopatra. Basma hanya tahu bahwa 15 indeks dikontrol oleh Cleopatra, namun ia tidak tahu indeks-indeks mana yang dikontrol.

Misalkan A adalah sebuah paket yang ingin Aisha kirim (yang kita sebut sebagai **paket orisinil**). Misalkan B adalah sebuah paket yang diterima oleh Basma (yang kita sebut sebagai **paket termanipulasi**). Untuk setiap i, sehingga $0 \leq i < 31$:

- jika Cleopatra tidak mengontrol bit di indeks i ($C[i] = 0$), maka Basma akan menerima bit i seperti yang dikirimkan oleh Aisha ($B[i] = A[i]$),
- jika tidak, yakni jika Cleopatra mengontrol bit di indeks i ($C[i] = 1$), maka nilai dari $B[i]$ akan ditentukan oleh Cleopatra.

Segera setelah mengirimkan sebuah paket, Aisha mengetahui apa isi dari paket termanipulasi.

Setelah Aisha mengirim seluruh paket, Basma akan menerima seluruh paket termanipulasi **sesuai urutan pengiriman** dan harus memulihkan pesan orisinil M.

Tugas Anda adalah menentukan dan mengimplementasikan sebuah strategi yang dapat Aisha gunakan untuk mengirimkan pesan M kepada Basma, sedemikian sehingga Basma dapat memulihkan M dari paket-paket termanipulasi. Secara spesifik, Anda harus

mengimplementasikan dua prosedur. Prosedur pertama akan melakukan langkah-langkah Aisha. Diberikan sebuah pesan M dan *array* C, kirim beberapa paket untuk mengirimkan pesan kepada Basma. Prosedur kedua akan melakukan langkah-langkah Basma. Diberikan paket-paket termanipulasi, pulihkan pesan orisinil M.

Detail Implementasi

Prosedur pertama yang harus Anda implementasikan adalah:

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

- M: sebuah array sepanjang S yang mendeskripsikan pesan yang Aisha ingin kirimkan ke Basma.
- C: sebuah array sepanjang 31 yang mendakan indeks-indeks yang dikontrol oleh Cleopatra.
- Prosedur ini bisa saja dipanggil **paling banyak 2100 kali** untuk setiap kasus uji.

Prosedur ini perlu memanggil prosedur berikut untuk mengirimkan sebuah paket:

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

- A: sebuah paket orisinil (sebuah *array* sepanjang 31) yang merepresentasikan bit-bit yang dikirim oleh Aisha.
- Prosedur ini mengembalikan paket termanipulasi B yang merepresentasikan bit-bit yang akan diterima oleh Basma.
- Prosedur ini dapat dipanggil paling banyak 100 kali untuk setiap pemanggilan `send_message`.

Prosedur kedua yang harus Anda implementasikan adalah:

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

- R: *array* yang mendeskripsikan paket-paket termanipulasi. Paket-paket berasal dari paket-paket yang dikirimkan oleh Aisha di satu pemanggilan `send_message` dan diberikan **sesuai urutan yang dikirimkan** oleh Aisha. Setiap elemen pada R adalah sebuah *array* sepanjang 31, yang merepresentasikan paket-paket termanipulasi.
- Prosedur ini harus mengembalikan sebuah *array* sepanjang S bit yang sama dengan pesan orisinal M.
- Prosedur ini bisa saja dipanggil **berkali-kali** untuk setiap kasus uji, **tepat satu kali** untuk setiap pemanggilan `send_message` yang bersesuaian. **Urutan dari pemanggilan prosedur** `receive_message` tidak harus sama dengan urutan pemanggilan `send_message` yang bersesuaian.

Perhatikan bahwa pada sistem *grading*, prosedur `send_message` dan `receive_message` akan dipanggil di **dua program yang terpisah**.

Batasan

- $1 \leq S \leq 1024$
- C terdiri dari tepat 31 elemen, yang 16 bitnya merupakan 0 dan 15 bitnya merupakan 1.

Subsoal dan Penilaian

Jika pada suatu kasus uji, pemanggilan prosedur `send_packet` tidak mematuhi aturan-aturan yang dijelaskan di atas, atau nilai keluaran dari suatu pemanggilan prosedur `receive_message` tidaklah tepat, maka nilai solusi Anda untuk kasus uji coba tersebut adalah 0.

Jika tidak, anggap Q adalah banyak pemanggilan prosedur `send_packet` maksimum dari seluruh pemanggilan `send_message` untuk seluruh kasus uji. Anggap juga X adalah sebagai berikut:

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

Maka, nilai akan dihitung sebagai berikut:

Subsoal	Nilai	Batasan Tambahan
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Tidak ada batasan tambahan.

Perhatikan bahwa di beberapa kasus, perilaku *grader* bisa saja **adaptif**, Artinya, nilai-nilai yang dikeluarkan oleh `send_packet` bisa saja bergantung tidak hanya pada parameter, tetapi juga pada hal-hal lain, seperti masukan dan nilai kembalian dari pemanggilan sebelumnya pada prosedur ini dan bilangan acak semu yang dihasilkan oleh *grader*. Perilaku *grader* **deterministik**, artinya jika Anda menjalankannya dua kali dan pada kedua pemanggilan Anda mengirimkan paket yang sama, ia akan memberikan perubahan yang sama

Contoh

Perhatikan pemanggilan berikut.

```
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])
```

Pesan yang Aisha coba kirimkan ke Basma adalah $[0, 1, 1, 0]$. Bit-bit dengan indeks dari 0 hingga 15 tidak bisa diganti oleh Cleopatra, sedangkan bit-bit dengan indeks dari 16 hingga 30 bisa diganti oleh Cleopatra.

Sebagai contoh saja, mari kita asumsikan Cleopatra mengisi bit yang dikontrolnya dengan 0 dan 1 secara bergantian, artinya ia memberikan 0 untuk indeks pertama yang ia kontrol (indeks 16 untuk kasus ini), 1 untuk indeks kedua yang ia kontrol (indeks 17), 0 untuk indeks ketiga yang ia kontrol (indeks 18), dan seterusnya.

Aisha bisa memutuskan untuk mengirim dua bit dari pesan orisinil dalam satu paket sebagai berikut: ia akan mengirim bit pertama di 8 indeks pertama yang ia kontrol dan bit kedua di 8 bit berikutnya yang ia kontrol.

Aisha kemudian memilih untuk mengirim paket berikut:

```
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])
```

Perhatikan bahwa Cleopatra bisa mengganti bit di 15 indeks terakhir, sehingga Aisha bisa memberikan mereka nilai apa saja, karena mereka bisa saja digantikan. Dengan strategi Cleopatra yang seperti diasumsikan, prosedur akan mengembalikan: $[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$.

Aisah memutuskan untuk mengirim dua bit terakhir dari M di paket kedua dengan cara yang serupa dengan sebelumnya:

```
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])
```

Dengan strategi Cleopatra yang seperti diasumsikan, prosedur akan mengembalikan: $[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$.

Aisa bisa mengirim paket yang lebih banyak, tetapi ia memutuskan untuk berhenti.

Grader kemudian memanggil prosedur berikut:

```
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 memulihkan pesan M sebagai berikut. Untuk setiap paket, ia mengambil bit pertama yang muncul dua kali berturut-turut, dan bit terakhir yang muncul dua kali berturut-turut. Dengan kata

lain, dari paket pertama, ia akan mengambil bit $[0, 1]$, dan dari paket kedua ia akan mengambil bit $[1, 0]$. Setelah itu, ia memulihkan pesan $[0, 1, 1, 0]$ yang merupakan nilai kembalian yang benar untuk pemanggilan ke `receive_message`.

Dapat ditunjukkan bahwa dengan strategi Cleopatra yang diasumsikan dan untuk semua pesan sepanjang 4, cara yang dipakai Basma memulihkan pesan M dengan benar, terlepas dari nilai C. Namun, cara ini tidaklah benar untuk kasus secara umum.

Contoh *Grader*

Contoh *grader* tidaklah adaptif. Tetapi, Cleopatra akan mengisi bit yang ia kontrol dengan bit 0 dan 1 secara bergantian, seperti yang dideskripsikan oleh contoh di atas.

Format masukan: **Baris pertama dari masukan mengandung sebuah bilangan bulat T, yang menyatakan banyaknya skenario.** Berikutnya diikuti dengan T skenario. Masing-masing skenario diberikan dengan format berikut:

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

Format keluaran: Contoh *grader* menulis hasil dari setiap T skenario dengan urutan yang sama yang diberikan oleh masukan dengan format berikut:

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

Di sini, K adalah banyaknya pemanggilan ke `send_packet`, D adalah pesan yang dikembalikan oleh `receive_message` dan L adalah panjangnya.