訊息

Aisha 和 Basma 是互相聯繫的朋友。 Aisha 有一串由 S 個位元 (即 0 或 1) 所構成的訊息 M ，預計傳送給 Basma。 Aisha 藉由傳送 **封包 (packets)** 來聯繫 Bashma。 一個封包是 31 個位元的序列，其編號由 0 到 30。 Aisha 想要藉由傳送若干個封包，來傳送訊息 M 給 Basma。

不幸的是，Cleopatra 會破壞 Aisha 和 Basma 之間的通訊，並且能夠 **偷改 (taint)** 封包的內容。 也就是在每一個封包， Cleopatra 可以更改 15 個位置的位元。 確切地說，也就是有一個長度為 31 的陣列 C，其中每一元素為 0 或 1，並具下列意義:

- $C[i] = 1$ 表示位置 (index) 為 i 的位元可被 Cleopatra 修改。我們稱這些位置被 Cleopatra **控制**。
- $C[i] = 0$ 表示位置 (index) 為 i 的位元不會被 Cleopatra 修改。

陣列 C 恰好包含 15 個 1 和 16 個 0。當傳送訊息時，被 Cleopatra 控制的封包位置是固定的。 Aisha 確切知道被 Cleopatra 所控制的 15 個位置。Basma 僅知道 Cleopatra 控制了 15 個位置，但她不知道確切的位置。

令 A 表示 Aisha 決定要傳送的一個封包 (我們稱之為 **初始封包**)。 令 B 表示會被 Basma 收到的封包 (我們稱之為 **毀損封包**)。 對每一個 i，若 $0 \le i < 31$:

- 如果 Cleopatra 沒有控制位置 i 的位元 ($C[i] = 0$)，則 Basma 收到位置 i 的位元為 Aisha 送的位元，即 ($B[i] = A[i]$)，
- 否則，如果 Cleopatra 控制了位置 i 的位元 ($C[i] = 1$)，則 $B[i]$ 的值將由 Cleopatra 決定。

在每傳送完一個封包後，Aisha立即知道被破壞的封包內容。

在 Aisha 傳送完所有封包後，Basma **按照傳送順序**收到所有損壞的封包，且必須還原出初始訊息 M。

你的任務是設計並實作一策略，使得 Aisha 能夠傳送訊息 M 給Basma，並使 Basma 能夠由損壞的封包還原出訊息M。 更明確地來說，你應該實作下列兩個程序。第一個程序將執行 Aisha 的動作。給定一個訊息 M 和陣列 C，該程序應該能夠藉由傳送若干個封包來傳送訊息給 Basma。 第二個程序將執行 Basma 的動作，給定損壞的封包，該程序應能夠還原初始訊息 M。

實作細節

第一個你應該實作的程序為:

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

- M: 一長度為 S 的陣列，為 Aisha 想要送給 Basma 的訊息。
- C: 一長度為 31 的陣列，表示 Cleopatra 所控制的位置。
- 針對每一筆測試資料，此程序可被呼叫**最多2100次**。

此程序應透過呼叫下列程序來傳送一個封包:

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

- A:一初始封包 (一個長度為 31 的陣列) 表示 Aisha 要傳送的一串位元。
- 此程序回傳一個損壞的封包 B，表示 Basma 將收到的一串位元。
- 每回呼叫`send_message`後，此程序最多可被呼叫 100 次。

第二個你應實作的程序如下:

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

- R:一描述毀損封包的陣列。 這些封包的原始來源是 Aisha 呼叫一次 `send_message` 所送的一些封包，且這些封包順序是按照Aisha **傳送的順序**。 陣列 R 的每一個元素為長度 31 的陣列，代表損壞的封包。

- 此程序應回傳 S 位元的陣列，內容即為初始訊息 M 。

- 針對每一筆測資，此程序可被呼叫**多次**；每次呼叫 `send_message` 時，此程序會被呼叫**恰好一次**。`receive_message`**程序呼叫**的 **順序**和對應的`send_message`呼叫順序未必一樣。

注意，評分系統中 `send_message` 程序和 `receive_message`程序由**兩個不同的程式**呼叫。

限制條件

- $1 \leq S \leq 1024$
- C恰好有 31 個元素，其中有 16 個等於 0 以及 15 個等於1。

子任務和評分

在任一筆測資，若呼叫`send_packet`程序不符合上述規則， 或任一次呼叫`receive_message`的回傳值不正確，則對該筆測資你的答案將評為 0 分。

否則，令 Q 表示對所有測資的所有呼叫 `send_message` 中，呼叫`send_packet`程序最多的次數。 同時令 X 等於:

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

則分數的計算方式如下:

Subtask	Score	Additional Constraints
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	No additional constraints.

注意在某些情形，評分程式的運算模式可以是**漸進式的(adaptive)**。這表示`send_packet`的回傳值會與輸入參數和之前呼叫此程序的回傳值相關。此外也會和評分程序的擬亂數產生方法有關。評分程序是**確定性的(deterministic)**的意思是如果你執行程式兩次並傳送同樣的封包，那將得到相同的變化。

範例

考慮下列呼叫。

```
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])
```

Aisha 嘗試傳送給 Basma 的訊息為 $[0, 1, 1, 0]$。Cleopatra無法改變的位元在位置 0 到15；Cleopatra可以改變的位元在位置16 到 30。

為方便解釋本範例，假設 Cleopatra 的行為模式為確定性的(deterministic)，且她在所控制位置以0 和 1 交替連續填寫，也就是她指定 0 到第一個她所控制的位置 (此例即為位置 16)， 1 到第二個她所控制的位置 (即位置 17)， 0 到第三個她所控制的位置 (即位置 18)， 依此類推。

Aisha可以決定用一個封包從初始訊息傳送兩個位元，如下: 她將在她所控制的前 8 個位置傳送第一個位元，並在接下來所控制的 8 個位置傳送第二個位元。

然後Aisha選擇傳送下列封包:

```
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])
```

注意 Cleopatra可以改變最後 15 個位置的位元，所以 Aisha 可以在這些位置設定任意的值，因為它們可能被更改。經由 Cleopatra 所假設的策略，該程序將回傳: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha決定用類似之前方式，使用第二個封包傳送 M 的最後兩個位元:

```
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])
```

藉由 Cleopatra 所設定的策略，該程序將回傳: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha可以傳送更多的封包，但她選擇不這樣做。

評分程式將執行下列程式呼叫:

```
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 還原訊息 M 的方式如下。從每一個收到的封包，她選取第一次連續出現兩次的位元，並且選取最後連續出現兩次的位元。 也就是，從第一個封包，她將選取位元 $[0,1]$，從第二個封包選取位元 $[1,0]$。 合併起來，她將還原訊息 $[0,1,1,0]$，這對呼叫`receive_message`是正確的回傳值。

當訊息長度為 4 時，依照Cleopatra的策略，不管 C 的值為何，可以證明 Basma 的方法可正確地還原 M。但是這正確性無法推廣到一般的情況。

樣本評分程序

樣本評分程序不是漸進式。Cleopatra的行為是確定性的 (deterministic)，且她以 0 和 1 交替連續填寫到她所控制的位置，如上述例子所述。

輸入格式: **第一列輸入一整數T，代表情境的個數.** 接下來有 T 種情境。 每一種情境的格式如下:

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

輸出格式: 樣本評分程序將每一情境的結果依照輸入的順序以下列格式輸出:

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

在此，K 表示呼叫 `send_packet` 的次數， D 是 `receive_message` 回傳的訊息，L 為其長度。