訊息

Aisha 和 Basma 是一對會互相聯繫的朋友。 Aisha 現在有一個訊息 M ，其中包含 B 個位元（只有 0 或 1）的序列，想要發送給 Basme。 Aisha 通過發送 **封包** 和 Basma 進行通訊。封包是一個包含了 31 位元的序列，索引為 0 到 30。 Aisha 想要透過發送一定數量的封包向 Basma 發送訊息 M 。

不幸地， Cleopatra 破壞了 Aisha 和 Basma 之間的交流並且能夠**污染**數據包。簡單來說，在每個封包中，Cleopatra 可以修改恰好 15 個位元。具體來說，現在有一個長度為 31 的數組 C ，其中每個元素要麼是 0 要麼是 1 ，含義如下：

- $C[i] = 1$ 表示索引為 i 的位元可能被 Cleopatra 更改。我們稱這些索引為 Cleopatra **可控制的**。
- $C[i] = 0$ 表示索引為 i 的位元不能被 Cleopatra 更改。

數組 C 恰好包含 15 個 1 和 16 個 0。 在傳送訊息時，Cleopatra 控制的索引集合對於所有封包保持不變。 Aisha 清楚知道哪 15 個索引是由 Clepatra 控制的。 Basma 只知道一共有 15 個索引由 Cleopatra 控制，但她不知道是哪些索引。

假設 A 是 Aisha 決定發送的封包（我們稱之為**原始封包**）。 令 B 為 Basma 收到的封包（我們稱之為**被污染封包**）。 對每個 i ，使得 $0 \le i < 31$ ：

- 如果 Cleopatra 不控制索引為 i 的位元（ $C[i] = 0$ ）， Basma 可以準確收到 Aisha 發送的位元 i ($B[i] = A[i]$)，
- 否則，如果 Cleopatra 控制索引為 i 的位元（ $C[i] = 1$ ）， $B[i]$ 的值由 Cleopatra 決定。

發送每個封包後， Aisha 會馬上得知對應的被污染封包的內容是什麼。

當 Aisha 發送完所有封包後， Basma 會依照**傳送順序**接收所有受污染的封包，並且需要重建原始訊息 M 。

你的任務是製定並實施一項策略，讓 Aisha 可以向 Basma 發送訊息 M ，隨後 Basma 從受污染的封包中恢復 M 。具體來說，您應該實施兩個子程式。第一個子程式執行 Aisha 的行動。該程式被給予一個訊息 M 和數組 C ，並且應該發送一些封包來將信息傳送給 Basma。第二個子程式執行 Basma 的操作。它被給予了被污染封包並且需要恢復原始訊息 M 。

實現細節

您應該實施的第一個子程式是：

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

- M： 長度為 S 的數組，給出 Aisha 想要傳送給 Basma 的訊息。
- C： 長度為 31 的數組，表示 Cleopatra 控制的位元的索引。
- 在每個測試案例中，此子程式最多會被調用**最多 2100 次**。

該子程式應該調用以下函數來發送封包：

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

- A：原始封包（長度為 31 的陣列） 代表 Aisha 發送的位元。
- 此子程式返回受污染的封包 B， 代表 Basma 將接收的位元。
- 此子程式最多可被調 100 次，在每次調用 `send_message` 時。

您應該實施的第二個子程式是：

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

- R： 描述受污染封包的數組。這些封包源自 Aisha 在 `send_message` 的調用中發送的封包，並按照 Aisha **發送的順序** 給出。 R 的每個元素都是一個長度為 31 的數組，代表一個受污染封包。
- 此程式應返回一個長度為 S 的數組。其與原始訊息 M 相同。
- 在每個測試案例中，此過程可能會被調用**多次**，對於每個對應的 `send_message` 調用**恰好一次**。`receive_message` 子程式 **調用的順序** 不一定與對應的`send_message`調用的順序相同。

請注意，在評分系統中， `send_message` 和`receive_message` 子程式在**兩個單獨的程式**中調用。

約束

- $1 \leq S \leq 1024$
- C 剛好有 31 元素，其中 16 個元素為 0 ， 15 個元素為 1。

子任務和評分

如果在任何測試案例中，對子程式 `send_packet` 的調用不符合上述規則， 或對子程式 `receive_message` 的任何調用的返傳值不正確，你對該測試案例的解決方案的得分將為 0。

否則，讓 Q 成為在所有測試案例中，子程式 `send_packet` 中對 `send_message` 的總調用次數的最大值。也讓 X 等於：

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

然後，分數的計算方法如下：

子任務	分數	附加約束
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	沒有其他限制。

請注意，在某些情況下，樣例評分程式的行為可以是**自適應的**。 這意味著 `send_packet` 返回的值可能不僅取決於它的輸入參數，還取決於許多其他因素， 包括先前呼叫此子程式的輸入和返回值，以及由評分程式中的分級機所產生的偽隨機數。分級機是**確定性的**，如果你運行兩次樣例評分程序並發送同樣的封包，它將會對它們進行相同的修改。

例子

考慮以下調用。

```
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 試圖向 Basma 發送的訊息是 $[0, 1, 1, 0]$ 。 索引從 0 到 15 的位元不能被 Cleopatra 更改，而索引從 16 到 30 的位元可以被Cleopatra 更改。

為了舉個例子，讓我們假設 Cleopatra 的行為是用交替的 0 和 1 填充她控制的連續位元，即她分配 0 到她控制的第一個索引（在我們的例子中是索引 16 ）， 分配 1 到她控制的第二個索引（索引 17 ）， 分配 0 到她控制的第三個索引（索引 18 ）， 等等。

Aisha 可以決定在一個封包中，傳送原始訊息中的兩位。如下所示：她將在她控制的前 8 索引中，發送原始信息的第一位，並且她控制的接下來 8 個索引處中，發送原始信息的第二位。

然後 Aisha 選擇發送以下封包：

```
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])
```

請注意，由於 Cleopatra 可以改變最後 15 索引的位元，所以 Aisha 可以任意設置它們，因為它們可能會被更改。 使用 Cleopatra 的假定策略，該過程返回：$[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]$ 。

Aisha 決定在第二個封包中發送 M 的最後兩位，與之前類似：

```
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])
```

使用 Cleopatra 的假定策略，該過程返回：$[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$。

Aisha 可以發送更多封包，但她選擇不這麼做。

然後樣例評分程式進行以下程式調用：

```
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 會依照下列方式還原訊息 M。 她從每個封包中取出連續出現兩次的第一個位元，以及連續出現兩次的最後一個位元。 也就是說，從第一個封包中，她取出位元$[0,1]$，從第二個封包中封包中她取出了位元$[1,0]$。把它們放在一起，恢復了訊息 $[0,1,1,0]$，這是對 `receive_message` 函數調用的正確回傳值。

可以證明，在假設的 Cleopatra 策略下，對於長度為 4 的訊息， Basma 的這種方法正確地恢復了 M，而不管 C 的值是多少。 但在一般情況下這並不正確。

樣例評分程式

樣例評分程式不具備適應性。相反地，在樣例評分程式中， Cleopatra 的行為是用交替的 0 和 1 位元填滿她控制的連續位，如上例所述。

輸入格式：**輸入第一行包含一個整數T， 指定場景的數量。** 接下來是 T 個場景。 它們中的每一個場境都採用以下格式進行描述：

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

輸出格式： 樣本評分員輸出每個 T 個場景的結果。按照以下格式輸入的順序輸出：

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

這裡， K 是調用 `send_packet` 的次數， D是 `receive_message` 回傳的訊息，且 L 是它的長度。