Aufgabe BinSearch

Eingabe	`stdin`
Ausgabe	`stdout`

```
bool binary_search(int n, int p[], int target){
    int left = 1, right = n;
    while(left < right){
        int mid = (left + right) / 2;
        if(p[mid] == target)
            return true;
        else if(p[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    if(p[left] == target) return true;
    else return false;
}
```

Wenn `p` sortiert ist, gibt dieser Code bekanntlich genau dann `true` zurück, wenn `target` in `p` vorkommt. Andererseits ist dies möglicherweise nicht der Fall, wenn `p` nicht sortiert ist.

Gegeben ist eine positive Ganzzahl n und eine Folge $b_1, \ldots, b_n \in \{\texttt{true}, \texttt{false}\}$. Es ist garantiert, dass $n = 2^k - 1$ für eine positive Ganzzahl k. Entwerfe eine Permutation p von $\{1, \ldots, n\}$, die folgende Bedingungen erfüllt. Sei $S(p)$ die Anzahl der Indizes $i \in \{1, \ldots, n\}$, für die `binary_search(n, p, i)` **nicht** b_i zurückgibt. Wähle p so, dass $S(p)$ klein ist (wie im Abschnitt "Limits" beschrieben).

(Anmerkung: Eine Permutation von $\{1, \ldots, n\}$ ist eine Folge von n Ganzzahlen, die jede Ganzzahl von 1 bis n *genau* einmal enthält.)

Eingabe

Die Eingabe enthält mehrere Testfälle. Die erste Zeile der Eingabe enthält T, die Anzahl der Testfälle. Es folgen die Testfälle.

Die erste Zeile eines Testfalls enthält die Ganzzahl n. Die zweite Zeile eines Testfalls enthält einen String der Länge n, der nur die Zeichen '0' und '1' enthält. Diese Zeichen sind nicht durch Leerzeichen getrennt. Falls das i-te Zeichen '1' ist, dann ist $b_i = \texttt{true}$ und falls es '0' ist, dann ist $b_i = \texttt{false}$.

Ausgabe

Die Ausgabe besteht aus den Antworten für jeden der T Testfälle. Die Antwort für einen bestimmten Testfall besteht aus der Permutation p, die für diesen Testfall erzeugt wurde.

Limits

- Sei $\sum n$ die Summe aller Werte von n in der Eingabe.
- $1 \leq \sum n \leq 100\,000$.
- $1 \leq T \leq 7\,000$.
- $n = 2^k - 1$ für ein $k \in \mathbb{N}$, $k > 0$.
- Ist $S(p) \leq 1$ für alle Testfälle innerhalb einer Teilaufgabe, dann erhält man 100% der Punkte für diese Teilaufgabe.
- Andernfalls, falls $0 \leq S(p) \leq \lceil \log_2 n \rceil$ (d.h. $1 < 2^{S(p)} \leq n + 1$) für alle Testfälle innerhalb einer Teilaufgabe, dann erhält man 50% der Punkte für diese Teilaufgabe.

#	Punkte	Limits
1	3	$b_i =$ `true`.
2	4	$b_i =$ `false`.
3	16	$1 \leq n \leq 7$.
4	25	$1 \leq n \leq 15$.
5	22	$n = 2^{16} - 1$ und jedes b_i wird unabhängig und mit gleicher Wahrscheinlichkeit nach dem Zufallsprinzip ausgewählt aus {`true`, `false`}.
6	30	Keine zusätzlichen Einschränkungen.

Beispiele

Eingabe	Ausgabe
4 3 111 7 1111111 3 000 7 000000000	1 2 3 1 2 3 4 5 6 7 3 2 1 7 6 5 4 3 2 1
2 3 010 7 0010110	3 2 1 7 3 1 5 2 4 6

Erläuterungen

Beispiel 1) In den ersten beiden Testfällen des ersten Beispiels haben wir $S(p) = 0$.

Im dritten Testfall haben wir $S(p) = 1$. Das liegt daran, dass `binary_search(n, p, 2)` `true` zurückgibt, obwohl $b_2 =$ `false`.

Im vierten Testfall haben wir $S(p) = 1$. Das liegt daran, dass `binary_search(n, p, 4)` `true` zurückgibt, obwohl $b_4 =$ `false`.

Beispiel 2) Wir haben $S(p) = 0$ für beide Testfälle.