Zadanie BinSearch

Wejście	`stdin`
Wyjście	`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;
}
```

Dobrze wiadomo, że jeśli `p` jest posortowane to powyższy kod zwraca `true` wtedy i tylko wtedy, gdy `target` znajduje się w `p`. Z drugiej strony, może to nie być prawdą jeżeli `p` nie jest posortowane.

Dana jest dodatnia liczba całkowita n oraz ciąg $b_1, \ldots, b_n \in \{\texttt{true}, \texttt{false}\}$. Gwarantowane jest, że $n = 2^k - 1$ dla pewnej dodatniej liczby całkowitej k. Musisz wygenerować permutację p zbioru $\{1, \ldots, n\}$ spełniającą pewne warunki. Niech $S(p)$ jest liczbą pozycji $i \in \{1, \ldots, n\}$ dla których `binary_search(n, p, i)` **nie** zwraca b_i. Twoim zadaniem jest wybrać p tak aby $S(p)$ było małe (jak opisano w sekcji "Ograniczenia").

(Uwaga: permutacja $\{1, \ldots, n\}$ to ciąg n liczb który zawiera każdą liczbę naturalną od 1 do n *dokładnie* raz.)

Wejście

Na wejściu podane jest wiele przypadków testowych. W pierwszym wierszu wejścia znajduje się liczba T, liczba przypadków testowych. Później następuje opis tych przypadków.

Pierwszy wiersz opisu zawiera liczbę naturalną n. Drugi wiersz opisu zawiera napis długości n składający się jedynie ze znaków `0` oraz `1` bez żadnych odstępów. Jeżeli i-ty znak napisu jest `1` to $b_i = \texttt{true}$, a jeżeli jest `0` to wtedy $b_i = \texttt{false}$.

Wyjście

Wyjście powinno zawierać odpowiedzi kolejno dla każdego z T przypadków testowych. Odpowiedź dla każdego testu to permutacja p wygenerowana dla tego przypadku.

Ograniczenia

- Niech $\sum n$ jest sumą wszystkich n (we wszystkich przypadkach testowych) w pojedynczym teście.
- $1 \leqslant \sum n \leqslant 100\,000$.
- $1 \leqslant T \leqslant 7\,000$.
- $n = 2^k - 1$ dla jakiegoś $k \in \mathbb{N}$, $k > 0$.
- Jeśli $S(p) \leqslant 1$ dla wszystkich przypadków testowych w podzadaniu to otrzymujesz 100% punktów za to podzadanie.
- W przeciwnym przypadku, jeżeli $0 \leqslant S(p) \leqslant \lceil \log_2 n \rceil$ (czyli $2 < 2^{S(p)} \leqslant n + 1$) dla wszystkich przypadków testowych w podzadaniu to otrzymujesz 50% punktów za to podzadanie.

#	Punkty	Ograniczenia
1	3	$b_i =$ `true`.
2	4	$b_i =$ `false`.
3	16	$1 \leqslant n \leqslant 7$.
4	25	$1 \leqslant n \leqslant 15$.
5	22	$n = 2^{16} - 1$, b_i są niezależnie i jednostajnie losowane z {`true`, `false`}
6	30	Brak dodatkowych ograniczeń.

Przykłady

Wejście	Wyjście
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

Wyjaśnienia

Przykład 1. W pierwszych dwóch przypadkach testowych $S(p) = 0$.

W trzecim przypadku $S(p) = 1$, ponieważ `binary_search(n, p, 2)` zwraca `true`, chociaż $b_2 =$ `false`.

W czwartym przypadku $S(p) = 1$, ponieważ `binary_search(n, p, 4)` zwraca `true`, chociaż $b_4 =$ `false`.

Przykład 2. $S(p) = 0$ dla obu przypadków.