ამოცანა BinSearch

Input file	stdin
Output file	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;
}
```

ცნობილია, რომ თუ p არის დასორტირებული, მოცემული კოდი აბრუნებს true-ს მაშინ და მხოლოდ მაშინ, თუ target მოიძებნება p მასივში. მეორე მხრივ, ეს შეიძლება არ იყოს ასე, თუ p არაა დასორტირებული.

თქვენ გადმოგეცემათ დადებითი მთელი რიცხვი n და მიმდევრობა $b_1, \ldots, b_n \in \{\texttt{true}, \texttt{false}\}$. გარანტირებულია, რომ $n = 2^k - 1$ რომელიღაც მთელი დადებითი k-სთვის. თქვენ უნდა შეადგინოთ $\{1, \ldots, n\}$-ის პერმუტაცია p, რომელიც დააკმაყოფილებს გარკვეულ პირობებს. აღვნიშნოთ $S(p)$-თი ისეთი $i \in \{1, \ldots, n\}$ ინდექსების რაოდენობა, რომლისთვისაც binary_search(n, p, i) არ აბრუნებს b_i-ს. თქვენ უნდა აარჩიოთ p ისე, რომ $S(p)$ იყოს პატარა (დეტალურად შეზღუდვების სექციაში ნახეთ).

(შენიშვნა: $\{1, \ldots, n\}$-ის პერმუტაცია არის რიცხვების მიმდევრობა, რომელიც 1-დან n-მდე ყველა რიცხვს *ზუსტად ერთხელ* შეიცავს.)

შესატანი მონაცემები

შემოსატანი მონაცემები რამდენიმე ტესტს შეიცავს. პირველ სტრიქონზე მოცემულია მთელი რიცხვი T - ტესტების რაოდენობა. შემდეგ შემოდის ტესტები თანმიმდევრობით.

ყოველი ტესტის პირველ სტრიქონში მოცემულია მთელი რიცხვი n. მეორე სტრიქონზე შემოდის n ზომის სტრიქონი, რომელიც შედგენილია მხოლოდ '0' და '1' სიმბოლოებით. ეს სიმბოლოები არ არიან ჰარით გამოყოფილი ერთმანეთისგან. თუ i-ური სიმბოლო არის '1', მაშინ $b_i = \texttt{true}$, და თუ არის '0', მაშინ $b_i = \texttt{false}$.

გამოსატანი მონაცემები

გამოსატანი მონაცემები შედგება მოცემული T ცალი ტესტის პასუხებისგან მიმდევრობით. თითოეული ტესტის პასუხი არის პერმუტაცია p, რომელსაც ამ შემთხვევისთვის დააგენერირებთ და დაბეჭდავთ ჰარებით გამოყოფილს.

შეზღდვები

- $\sum n$-ით აღვნიშნოთ მოცეულ ტესტებში ყველა n-ის მნიშვნელობის ჯამი.
- $1 \leq \sum n \leq 100\,000$.
- $1 \leq T \leq 7\,000$.
- $n = 2^k - 1$, სადაც $k \in \mathbb{N}$, $k > 0$.
- თუ $S(p) \leq 1$ ქვეამოცანის ყველა ტესტისთვის, მაშინ თქვენ მიიღებთ ქვეამოცანისთვის განკუთვნილი ქულის 100%-ს.

- წინააღმდეგ შემხვევაში, თუ $0 \leq S(p) \leq \lceil \log_2 n \rceil$ (i.e. $1 \leq 2^{S(p)} \leq n+1$) ქვეამოცანის ყველა ტესტისთვის, მაშინ თქვენ მიიღებთ ქვეამოცანისთვის განკუთვნილი ქულის 50%-ს.

#	ქულა	შეზღდვები
1	3	$b_i = \texttt{true}$.
2	4	$b_i = \texttt{false}$.
3	16	$1 \leq n \leq 7$.
4	25	$1 \leq n \leq 15$.
5	22	$n = 2^{16} - 1$ და b_i-ები აირჩევა თანაბარალბათურად და დამოუკიდებლად $\{\texttt{true}, \texttt{false}\}$-
6	30	დამატებით შეზღუდვები არ არის.

მაგალითები

Input file	Output file
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

განმარტებები

განმარტება 1. პირველი მაგალითის პირველ ორ ტესტში გვაქვს $S(p) = 0$.

მესამე ტესტში გვაქვს $S(p) = 1$, რადგან `binary_search(n, p, 2)` აბრუნებს `true`-ს, თუმცა $b_2 = \texttt{false}$.

მეოთხე ტესტში გვაქვს $S(p) = 1$, რადგან `binary_search(n, p, 4)` აბრუნებს `true`-ს, თუმცა $b_4 = \texttt{false}$.

განმარტება 2. მეორე მაგალითში ორივე ტესტისთვის გვაქვს $S(p) = 0$.