F. Maximize Nor

time limit per test: 4 seconds
memory limit per test: 1024 megabytes

The bitwise nor* of an array of k-bit integers $b_1, b_2, \ldots, b_m$ can be computed by calculating the bitwise nor cumulatively from left to right. More formally, $\operatorname{nor}(b_1, b_2, \ldots, b_m) = \operatorname{nor}(\operatorname{nor}(b_1, b_2, \ldots, b_{m-1}), b_m)$ for $m \geq 2$, and $\operatorname{nor}(b_1) = b_1$.

You are given an array of k-bit integers $a_1, a_2, \ldots, a_n$. For each index i ($1 \leq i \leq n$), find the maximum bitwise nor among all subarrays† of a containing index i. In other words, for each index i, find the maximum value of $\operatorname{nor}(a_l, a_{l+1}, \ldots, a_r)$ among all $1 \leq l \leq i \leq r \leq n$.

* The logical nor of two boolean values is 1 if both values are 0, and 0 otherwise. The bitwise nor of two k-bit integers is calculated by performing the logical nor operation on each pair of the corresponding bits.

For example, let us compute $\operatorname{nor}(2, 6)$ when they are represented as 4-bit numbers. In binary, $2{=}0010_2$ and $6 = 0110_2$. Therefore, $\operatorname{nor}(2,6) = 1001_2 = 9$ as by performing the logical nor operations from left to right, we have:

- $\operatorname{nor}(0,0) = 1$
- $\operatorname{nor}(0,1) = 0$
- $\operatorname{nor}(1,0) = 0$
- $\operatorname{nor}(1,1) = 0$

Note that if 2 and 6 were represented as 3-bit integers instead, then $\operatorname{nor}(2,6) = 1$.

† An array x is a subarray of an array y if x can be obtained from y by the deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($1 \leq n \leq 10^5, 1 \leq k \leq 17$) — the number of elements in the array and the number of bits of the array elements.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 2^k - 1$) — the elements of array a.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output n integers, the i-th of which is the maximum bitwise nor among all subarrays of a containing index i.

Example

input Copy

```
2
2 2
1 3
5 3
1 7 4 6 2
```

output Copy

```
1 3
5 7 5 6 5
```

Note

In the first test case, subarrays that have index 1 are $[1]$ and $[1, 3]$. The values of their bitwise nor are 1 and 0 respectively. Hence, the answer for index 1 is 1. Subarrays that have index 2 are $[3]$ and $[1, 3]$. The values of their bitwise nor are 3 and 0 respectively. Hence, the answer for index 2 is 3.

In the second test case:

- For $i = 1$, the subarray with maximum bitwise nor is $[a_1, a_2, a_3, a_4, a_5] = [1, 7, 4, 6, 2]$, $\operatorname{nor}(1, 7, 4, 6, 2) = 5$
- For $i = 2$, the subarray with maximum bitwise nor is $[a_2] = [7]$, $\operatorname{nor}(7) = 7$

Codeforces Round 1019 (Div. 2)
Finished
Practice

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++23 14.2 (64 bit, ms

Choose file: Choose File No file chosen

Submit

→ Last submissions

Submission	Time	Verdict
316663807	Apr/22/2025 13:15	Accepted

→ Problem tags

bitmasks data structures dp
implementation

No tag edit access

→ Contest materials

- Announcement (en)
- Tutorial (en)

- For $i = 3$, the subarray with maximum bitwise nor is $[a_1, a_2, a_3, a_4, a_5] = [1, 7, 4, 6, 2]$, $\mathrm{nor}(1, 7, 4, 6, 2) = 5$
- For $i = 4$, the subarray with maximum bitwise nor is $[a_4] = [6]$, $\mathrm{nor}(6) = 6$
- For $i = 5$, the subarray with maximum bitwise nor is $[a_1, a_2, a_3, a_4, a_5] = [1, 7, 4, 6, 2]$, $\mathrm{nor}(1, 7, 4, 6, 2) = 5$