

Day 83 coding Statement :

Given a complete binary tree with the height of H , we index the nodes respectively top-down and left-right from 1. The i -th node stores a positive integer V_i . Define P_i as follows: $P_i = V_i$ if the i -th node is a leaf, otherwise $P_i = \max(V_i * P_L, V_i * P_R)$, where L and R are the indices of the left and right children of i , respectively. Your task is to calculate the value of P_1 .

Input

There are several test cases (fifteen at most), each formed as follows:

- The first line contains a positive integer H ($H \leq 15$).
- The second line contains $2^H - 1$ positive integers (each having a value of 10^9 at most), the i -th integer shows the value of V_i .

The input is ended with $H = 0$.

Output

For each test case, output on a line an integer which is the respective value of P_1 found, by modulo of 1,000,000,007.

Sample Input

```
2
1 2 3
3
3 1 5 2 6 4 7
0
```

Sample Output

```
3
105
```

Explanation

```
  3
 / \
/   \
```

1 5

/\ /\

2 6 4 7

```
import java.math.BigInteger;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

class Main {
    public static BigInteger MOD = new BigInteger("1000000007");

    public static BigInteger pr(int i, int len, BigInteger v[]) {
        if (2 * i > len)
            return v[i];
        return pr(2 * i, len, v).max(pr(2 * i + 1, len, v)).multiply(v[i]);
    }

    public static void main(String[] ar) throws IOException {
        BufferedReader br = new BufferedReader(new
        InputStreamReader(System.in));
        int n, len;
        BigInteger v[];
        String tmp[];
        while ((n = Integer.parseInt(br.readLine())) != 0) {
            len = (1 << n) - 1;
            v = new BigInteger[len + 5];
            tmp = br.readLine().split(" ");
            for (int i = 1; i <= len; i++)
                v[i] = new BigInteger(tmp[i - 1]);
            System.out.println(pr(1, len, v).mod(MOD));
        }
    }
}
```