Problem

We consider a number to be *beautiful* if it consists only of the digit 1 repeated one or more times. Not all numbers are beautiful, but we can make any base 10 positive integer beautiful by writing it in another base.

Given an integer **N**, can you find a base *B* (with *B* > 1) to write it in such that all of its digits become 1? If there are multiple bases that satisfy this property, choose the one that maximizes the number of 1 digits.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with an integer **N**.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the base described in the problem statement.

Limits

1 ≤ **T** ≤ 100.

Small dataset

3 ≤ **N** ≤ 1000.

Large dataset

3 ≤ **N** ≤ 1018.

Sample

|  |  |
| --- | --- |
| Input | Output |
| 2  3  13 | Case #1: 2  Case #2: 3 |

In case #1, the optimal solution is to write 3 as 11 in base 2.

In case #2, the optimal solution is to write 13 as 111 in base 3. Note that we could also write 13 as 11 in base 12, but neither of those representations has as many 1s.

题意：给出一个数，问要以全1表示，最小的进制是多少。比如3以2进制表示是11 全1表示，答案是2。

思路：给出的数n是十进制，不论是几进制的表示法，都可以用以下公式得到数n

1 + base + base^2 + base^3+...+base^(bit-1) = n

base是进制数，bit是全1表示的总位数

首先要发现一件事，任意一个数n都可以用 n-1进制数表示为11，所以，任何一个数（因为N>=3没有0的情况）

都是有答案的。

又因为上面这个公式所以可以用搜索得到答案。我们可以枚举位数bit，位数最长就是进制最小的时候，

进制最小是2，长度是64，所以从最长的位数依次枚举就可以了。

当位数一定时，上式就得到了一个函数，比如bit等于3时 函数为  y = 1 + x + x^2，

现在就可以把问题变换为这个函数值为n的时候是否存在整数解，而这又是一个单调递增的函数。

所以可以用二分搜索快速求出是否有解。

**[cpp]** [view plain](http://blog.csdn.net/xishisugan/article/details/54836602) [copy](http://blog.csdn.net/xishisugan/article/details/54836602)

1. #include <cstdio>
2. **typedef** **long** **long** ll;
3. **using** **namespace** std;
4. ll calc(ll b, **int** bit, ll x)
5. {
6. ll ret = 1;
7. **for** (**int** i = 1; i < bit; i++)
8. {
9. **if** (x/b >= ret)          //以免相乘直接溢出64位整数，所以做这样的处理
10. {
11. ret \*= b;
12. ret++;
13. }**else**
14. {
15. ret = x + 1;        //大于的时候只需要
16. **break**;              //传递大于n的信息即可，所以直接赋值n+1即可
17. }
18. }
19. **return** ret;
20. }
21. **bool** judge(ll x, **int** bit, ll &base)
22. {
23. ll l = 2, r = x;
24. **while** (l < r)
25. {
26. ll mid = (l + r)>>1;
27. **if** (calc(mid, bit, x) < x)
28. l = mid + 1;
29. **else**
30. r = mid;
31. }
32. base = l;
33. **return** calc(base, bit, x) == x;
34. }
35. **void** run()
36. {
37. ll n;
38. ll base;
39. scanf("%lld", &n);
40. base = n - 1;
41. **for** (**int** i = 60; i >= 2; i--)
42. **if** (judge(n, i, base))
43. **break**;
44. printf("%lld\n", base);
45. }
46. **int** main()
47. {
48. **int** T, cas = 1;
50. scanf("%d", &T);
51. **while** (T--)
52. {
53. printf("Case #%d: ", cas++);
54. run();
55. }
56. **return** 0;
57. }