**Find position of set bit**

[bit](http://www.practice.geeksforgeeks.org/tag-page.php?tag=bit&isCmp=0)[Microsoft](http://www.practice.geeksforgeeks.org/tag-page.php?tag=Microsoft&isCmp=1)

Given a number having only one ‘1’ and all other ’0’s in its binary representation, find position of the only set bit. If there is only one '1' bit then print that position else print -1. Position of  set bit '1' should be counted starting with 1 from LSB side in binary representation of the number.

**Input:**

The first line of input contains an integer T denoting the number of test cases.  
The first line of each test case is N.  
  
**Output:**

Print the position if there is only one '1' bit, else print -1. For each test case print the output in a new line.  
  
**Constraints:**

1 ≤ T ≤ 100  
1 ≤ N ≤ 20000000  
  
**Example:**

**Input**  
2  
2  
5

**Output**  
2  
-1

\*\*For More Examples Use Expected Output\*\*

<http://www.practice.geeksforgeeks.org/problem-page.php?pid=638>

#include <iostream>

#include <stdio.h>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int t;

scanf("%d", &t);

while(t--) {

int n;

scanf("%d", &n);

std::vector<int> v;

while(n > 0) {

v.push\_back(n%2);

n/=2;

}

std::reverse(v.begin(),v.end());

bool hayuno= false;

int pos = 0;

for(int i =0; i < v.size(); i++) {

//printf("%d", v[i]);

if(v[i] == 1) {

if(!hayuno) {

pos = i;

}else {

pos = -1;

break;

}

hayuno = true;

}

}

if(pos != -1) {

printf("%d\n", v.size() - pos);

}else{

printf("%d\n",-1);

}

}

return 0;

}