Module 1 : Programming

Be seated : 3 points

Buddy Allocation : 3 points

Siri Transform : 7 points

Marbles : 7 points

BE SEATED

Problem statement :

Jimmy was in the metro this morning travelling from start to end of the red line. Tired, he desperately wanted a seat. He noticed that there were 7 seats in front of him . He knew all the passengers seated and was sure that 5 of them would go with him till the last station. He also knew that there were 50% chances of each of the other guys getting up . He did a quick calculation and figured out his chances of getting a seat at the next stop. He also figured out what would be his chances in case there were n seats , and not only 2.

Let us see what he worked out :

let the 2 gentlemen be A and B. Let p(A) be the probability that A gets up on the next stop.

Given , p(A)=.5 p(B)=.5

probability that none of them gets up = pbar(A)*pbar(B)=.5*.5=.25;

probability that atleast one of them gets up =probablitiy that he gets a seat=1-.25= .75

if there were n seats, and the probability that the person sitting on each seat would get up at the next stop is p, then , probability of getting a seat :

$$P(SEAT)= 1-((p')^n)$$

Theoretically, we need infinite seats to be sure that P(SEAT)=1 . But the precision handling capabilities of prgramming languages are not infinite! So there is a practical limit on the value of n after which the value of P(SEAT) becomes 1. For example, here is a sample run with p=1/2.

n=1 P_SEAT=0.500000
n=2 P_SEAT=0.750000
n=3 P_SEAT=0.875000
n=4 P_SEAT=0.937500
n=5 P_SEAT=0.968750
n=6 P_SEAT=0.984375
n=7 P_SEAT=0.992188
n=8 P_SEAT=0.996094
n=9 P_SEAT=0.998047
n=10 P_SEAT=0.999023
n=11 P_SEAT=0.999512
n=12 P_SEAT=0.999756
n=13 P_SEAT=0.999878
n=14 P_SEAT=0.999939
n=15 P_SEAT=0.999969

n=16 P_SEAT=0.999985
n=17 P_SEAT=0.999992
n=18 P_SEAT=0.999996
n=19 P_SEAT=0.999998
n=20 P_SEAT=0.999999
n=21 P_SEAT=1.000000
n=22 P_SEAT=1.000000
n=23 P_SEAT=1.000000
n=24 P_SEAT=1.000000
n=25 P_SEAT=1.000000
p_SEAT =1 after 25 seats

(Note that the output specifies the number of seats as 25, even though the actual number is 21, this is because floating point mathematics is unexact. However , you need to display only the value when == returns true)

Input :
The first line would be an integer N, the number of test cases, followed by exactly N lines, each specifying the value p, the probability of the person getting up for that case.

Output :

Exactly N lines, each specifying the number of seats required to be sure to get a seat .

Sample :

3
.5
.25
.75

Output :
[aman@aman c]$ ./SeatAvailable
3
.5

p_SEAT =1 after 25 seats for p=0.500000
.25

p_SEAT =1 after 61 seats for p=0.250000
.75

p_SEAT =1 after 13 seats for p=0.750000

Buddy Allocation

Problem Statement

Rishab and Rakshit decide that it is time they try writing an operating system. Not surprisingly , they have chosen to follow a very strange memory allocation policy for their system.
The policy is as follows :

Whenever a process demands N K (Kilobytes=1024 bytes) blocks of memory, the system, instead of allocating a block of N bytes, allots a block which is of the size equal to nearest higher power of 2.

For eg, if a request comes for a 6K block, system will allot a memory block of 8K, similarily, a request for a 33K block shall be satisfied with a block of 64K.

Help them implement this scheme!

Input :

First line N, number of requests , followed by N requests. Each request can be contained in 32 bit numbers.

Output :

The amount of memory actually allotted for each request (assume every request is satisfied) .
It is guaranteed that no allocation needs to be greater than 2^31;

Sample :
Input :
4

1K
13K
99K
343948394K

Output :

1K
16K
128K
536870912K

SIRI TRANSFORM

Problem Statement

After the ios6 update, Ankur and Bibhas were hoping that perhaps their lonliness would be cured by siri. Alas! This was not meant to be as their thick Indian accent ensured that Siri wouldn't understand a word of what they so meliflously spoke to her.
Disappointed, Ankur and Bibhas started looking for ways to fix the siituation and endeds up inventing the following transformations, that when applied to strings, will make them understandable to siri(well, atleast according to them).

Let C be a character of the string Z, upon which the siri transform has to be applied.

1. if C belongs to the set of vowels ,(a,e,i,o,u),, then replace C with CCC (repeat it).
2. If C is a character but does not belongs to the set of vowels, then replace C with CC.
3. Leave all the other characters as it is.
4. Rearrange the string with the last word appearing first (It's their invention, Don't Blame us).

Input

N , the number of lines, followed by N lines, each having a string made up of lowercase characters and other special characters.

Assume that no word would be of length greater than 10 , and there are atmost 10 words in a line.

Output

N lines, each displaying the siri transform of respective inputs.

Sample

3
siri play rock
siri find coffee
siri call me vaio

rrooocckk ppllaaayy ssiiirriii
ccoooffffeeeeee ffiiinndd ssiiirriii
vvaaaiiiooo mmeee ccaaallll ssiiirriii

MARBLES

Problem Statement :

Rohit dreams he is in a shop with an infinite amount of marbles. He is allowed to select n marbles. There are marbles of k different colors. From each color there are also infinitely many marbles. Rohit wants to have at least one marble of each color, but still there are a lot of possibilities for his selection. In his effort to make a decision he wakes up.
Now he asks you how many possibilities for his selection he would have had. Assume that marbles of equal color can't be distinguished, and the order of the marbles is irrelevant.

Input

The first line of input contains a number T <= 100 that indicates the number of test cases to follow. Each test case consists of one line containing n and k, where n is the number of marbles Rohit selects and k is the number of different colors of the marbles. You can assume that 1<=k<=n<=1000000.

Output

For each test case print the number of possibilities that Rohit would have had. You can assume that this number fits into a signed 64 bit integer.

Example

Input:
2
10 10
30 7

Output:
1
475020

Module 2 : Bit Hacks

Instructions :

1. All the questions carry equal weightage
2. The Bit calculator is there to help you out.
3. ASSUME ALL THE NUMBERS ARE 8 BIT UNSIGNED NUMBERS.
4. For reference, the following table may be helpful :

Decimal	Binary
255	11111111
254	11111110
252	11111100
248	11111000
240	11110000
224	11100000
192	11000000
128	10000000
0	00000000

And : &

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

OR : |

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

XOR : A ^ B

A	B	A ^ B

0	0	0
0	1	1
1	0	1
1	1	0

Not : ~

A	~A
0	1
1	0

Left shift : >>
Eg : 101 >>1 gives 010
Right Shift : <<
Eg : 110 <<1 gives 100
> Whenever sign is involved, Assume 2's complement arithmetic. In case you don't know this , just remember , in 2s complement arithmetic :

-1 : 1111111111 . Stating in english , -1 is all 1s.

Just to repeat :

```
&   -  bitwise and
|   -  bitwise or
^   -  bitwise xor
~   -  bitwise not
<<  -  bitwise shift left
>>  -  bitwise shift right
```

Ready ? Let us start!

Assume all lower case letters are binary variables.

Q 0001 :

(a ^ a)=

Q 0010 :

(a^255)=_____

Q 0011 :

((4 | 5)& (6 | 7)) & 0 =_____

Q 0100 :

(128 | 64 | 32 | 16 | 8 | 4 | 2 | 1) =

Q 0101 :

(128 & 64 & 32 & 16 & 8 & 4 & 2 & 1)=

Q 0110 :

4^((4^5)&-(4<5))=

Q0111:

5^((4^5)&-(4<5))=

Q1000: What does the following yields :

$log_2(a \ \& \ -a)$?

Q1001:

12 << 4= ?

Q1010 :

12>>2=?

Q1011 :

(((143&231)&(~(223)||(23^34))||(343^343))&123

tip : Remeber the command line!

Q1100: What does the following imply :

X & (X – 1) == 0 returns true.

Q1101 : if x,y are integers, what would the following do :

```
x=x^y;
y=x^y;
x=x^y;
```

Q1110:

To set the nth bit of a number, which of the following would work :

```
X=X|(1<<n);
X=X|~(1<<n);
```

Q1111:

Write the binary representation of first 14 fibonnaci numbers.
(0,1,1,...)