DO NOT POST YOUR CODE IN PIAZZA.

Read all the questions carefully. You can ask for clarification in piazza, but not for answers. You can search over the internet to learn, but must not ask questions to any online forum except for piazza.

All your programs will be checked by a program. Therefore, please strictly follow the instructions.

All the codes will be checked with a plagiarism checker, and any case of plagiarism, if detected, will be reported to the department.

If your overall score is below 30% in this assignment, then you might be considered for some manual partial marking, but that will not happen before the midterm.

Do not worry about the time limit, but focus only on the complexity. The time limit are for the instructors so that they can set up an appropriate automated testing environment.

A. Palindrome (Time limit: 10 seconds)

Problem Description

A palindromic number is an integer that is the same when the digits are reversed. For example, 121 and 625526 are palindromic, but 625 is not a palindromic number.

Input:

The input is in 'palindrome.txt'. The first line of the input contains the line count m ($1 \leq m \leq 1,000$), which is the number of lines that follows the first line. Each of the following lines contains an integer of n digits ($1 \leq n \leq 5,000$).

Sample input:

```
3.
12333.
92837465.
1000000000000000123.
```

Output:

The output consists of m lines. Each line transforms the corresponding input line into a palindrome by concatenating the input and its digits in reversed order. To minimize the output length, you must not repeat the trailing digit (or, digits if there are multiple occurrences of the same digit) of the input integer.

Sample output:

```
1233321.
928374656473829.
10000000000000001232100000000000000001.
```

Submission and Grading Information

1. The expected complexity is $O(n)$.
2. Your can assume that the input integers are positive, and does not start with zero. Write a code 'A.YourNSID.java' that reads 'palindrome.txt' and writes the output in a file called 'A.YourNSID.txt'. **Only submit the file 'A.YourNSID.java' in Moodle.**
3. The score will be 0 if your program does not terminate within 10 seconds. If your output is correct for k out of m inputs, then you will receive a score of $\lceil \frac{k}{m} \cdot 10 \rceil$.

> Every palindrome of 4 digits is divisible by 11 (good to know, but not related to this assignment). Can you prove it in a midterm question?

B. Beer Time (Time limit: 3 seconds)

Problem Description

A group of n students of CMPT360 is standing around a circle, but there are $(n-1)$ bottles of water and only 1 bottle of beer. They came up with an algorithm to solve their problem (yes, this is how those students are). They will start counting clockwise, starting at position 1, removing every second remaining person to get out of the circle and have a bottle of water. As the process goes on, the circle becomes smaller and smaller, until only one lucky student remains, who can have that only bottle of beer. A smart student quickly computes the position $J(n)$ that would get the bottle of beer and stand in that position.

For example, if there are 10 people numbered 1 to 10 in clockwise order around the circle, then the order of getting a water bottle is 2, 4 , 6, 8, 10, 3, 7, 1, 9, as follows (* indicates the position where the counting starts):

```
*1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (initial state)
1, *3, 4, 5, 6, 7, 8, 9, 10
1, 3, *5, 6, 7, 8, 9, 10
1, 3, 5, *7, 8, 9, 10
1, 3, 5, 7, *9, 10
*1, 3, 5, 7, 9
1, *5, 7, 9
1, 5, *9
*5, 9
```

```
*5 (survivor)
```

Here the person at the 5th position survives. Therefore, $J(10) = 5$. A quick way to find the $J(n)$ is to use the following:

$$J(1) = 1$$
$$J(2^m + x) = 2x + 1,$$

where $m \geq 0$ and $0 \leq x < 2m$.

For example, if $n = 10$, then $J(n) = J(10) = J(2^3 + 2)$. Therefore in this case, $m = 3$ and $x = 2$, and hence $J(10) = 2 * 2 + 1 = 5$. Similarly, if $n = 16$, then $J(n) = J(16) = J(2^4 + 0)$. Therefore, $m = 4, x = 0$, and $J(16) = 2 * 0 + 1 = 1$.

Input:

The input is in 'beer.txt'. The first line of the input contains the line count c ($1 \leq c \leq 1,000$), which is the number of lines that follows the first line. Each of the following lines contains an integer n ($1 \leq n \leq 10,000$).

Sample input:

```
2.
16.
10.
```

Output:

The output consists of c lines. Each line prints the lucky position $J(n)$ of the corresponding input.

Sample output:

```
1.
5.
```

Submission and Grading Information

1. The expected complexity is $O(\log n)$.
2. Write a code 'B.YourNSID.java' that reads 'beer.txt' and writes the output in a file called 'B.YourNSID.txt'. **Only submit the file 'B.YourNSID.java' in Moodle.**
3. The score will be 0 if your program does not terminate within 3 seconds. If your output is correct for k out of m inputs, then you will receive a score of $\lceil \frac{k}{m} \cdot 10 \rceil$.

C. Recursion (Time Limit: 3 Seconds)

Recursion may appear in various contexts and in different forms. For fast implementation, we should always aim at transforming recursions into a simpler form of computation. In this assignment, the task is to evaluate $X(\cdot)$, which is defined as follows:

$$X(m,n) = \begin{cases} 0, \text{if } m = 0 \text{ or } n = 0 \\ X(m, n-1), \text{if } n \text{ is odd and } m \text{ is even} \\ X(m-1, n), \text{if } m \text{ is odd and } n \text{ is even} \\ X(m-1, n-1) + m + n, \text{otherwise.} \end{cases}$$

Input:

The input is in 'recursion.txt'. The first line of the input contains the line count c ($1 \leq c \leq 1,000$), which is the number of lines that follows the first line. Each of the following lines contains a pair of integers m, n ($1 \leq m, n \leq 10,000$).

Sample input:

```
3.
102,21.
10000,10000.
```

Output:

The output consists of c lines. Each line prints $X(m, n)$.

Sample output:

```
2060.
100010000.
```

Submission and Grading Information

1. The expected complexity is $O(1)$.
2. Write a code 'C.YourNSID.java' that reads 'recursion.txt' and writes the output in a file called 'C.YourNSID.txt'. **Only submit the file 'C.YourNSID.java' in Moodle.**
3. The score will be 0 if your program does not terminate within 3 seconds. If your output is correct for k out of m inputs, then you will receive a score of $\lceil \frac{k}{m} \cdot 10 \rceil$.

D. Software Testing (Time Limit: 10 Seconds)

Murphy's law states that 'things will go wrong in any given situation, if you give them a chance,' or more commonly, 'whatever can go wrong, will go wrong.' Many automated software testing tries to test for all possible inputs, whenever the number of possibilities are reasonably small.

An image is normally processed by a set of three satellite modules m_1, m_2, m_3 in this order. The image might get corrupted during the transfer. During this process, a module m_j can request any of its predecessor m_i, where $1 \leq i < j \leq 3$, to resend the information. Upon receiving a resend request, m_i must pass the data to the next module m_{i+1}.

A normal processing must start at m_1 and end at m_3. If the maximum number of total resend requests exceeds n, then the system generates an alarm.

We want to count the number of different ways with exactly n resend requests. If $n = 1$, then there are exactly 3 different ways (* denotes the resend request).

```
12*123  [2 gets the corrupt data and decides to ask from 1]
123*23  [3 gets the corrupt data and decides to ask from 2]
123*123 [3 gets the corrupt data and decides to ask from 1]
```

If $n = 2$, then the number of different ways is 8.

```
12*12*123
12*123*123
12*123*23
123*23*23
123*23*123
123*123*23
123*123*123
123*12*123
```

Input:

The input is in 'software.txt'. The first line of the input contains the line count m ($1 \leq m \leq 1,000$), which is the number of lines that follows the first line. Each of the following lines contains an integer n ($1 \leq n \leq 10,000$).

Sample input:

```
2.
1.
2.
```

Output:

The output consists of m lines. Each line prints the number of different ways that exactly n resend requests can happen.

Sample output:

```
3.
8.
```

Submission and Grading Information

1. The expected complexity is $O(n)$.
2. Write a code 'D.YourNSID.java' that reads 'software.txt' and writes the output in a file called 'D.YourNSID.txt'. **Only submit the file 'D.YourNSID.java' in Moodle.**
3. The score will be 0 if your program does not terminate within 10 seconds. If your output is correct for k out of m inputs, then you will receive a score of $\lceil \frac{k}{m} \cdot 10 \rceil$.

E. World of Unknowns, Time limit: 3 seconds

The Collatz conjecture is about a sequence: start with any positive integer n. Then each term is obtained from the previous term as follows: if the previous term is even, the next term is one half the previous term. If the previous term is odd, the next term is 3 times the previous term plus 1. The conjecture is that no matter what value of n, the sequence will always reach 1.

$$f(n) = \begin{cases} 1, \text{if } n = 1 \\ f(3n+1), \text{if } n \text{ is odd} \\ f(n/2), \text{otherwise.} \end{cases}$$

Your task is to count the number of terms in the sequence. For example, if $n = 12$, then the sequence has 10 terms: $12, 6, 3, 10, 5, 16, 8, 4, 2, 1$.

Input:

The input is in 'sequence.txt'. The first line of the input contains the line count m ($1 \leq m \leq 1,000$), which is the number of lines that follows the first line. Each of the following lines contains an integer n ($1 \leq n \leq 10,000$).

Sample input:

```
2.
12.
27.
```

Output:

The output consists of m lines. Each line prints the number of terms in the sequence of $f(n)$.

Sample output:

```
10.
112.
```

Submission and Grading Information

1. The complexity is unknown, but try your fastest algorithm :)
2. Write a code 'E.YourNSID.java' that reads 'sequence.txt' and writes the output in a file called 'E.YourNSID.txt'. **Only submit the file 'E.YourNSID.java' in Moodle.**
3. The score will be 0 if your program does not terminate within 10 seconds. If your output is correct for k out of m inputs, then you will receive a score of $\lceil \frac{k}{m} \cdot 10 \rceil$.