Default Arguments

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

Python supports a useful concept of default argument values. For each keyword argument of a function, we can assign a default value which is going to be used as the value of said argument if the function is called without it. For example, consider the following increment function:

```
def increment_by(n, increment=1):
    return n + increment
```

The functions works like this:

```
>>> increment_by(5, 2)
7
>>> increment_by(4)
5
>>>
```

Debug the given function `print_from_stream` using the default value of one of its arguments.

The function has the following signature:

```
def print_from_stream(n, stream)
```

This function should print the first n values returned by `get_next()` method of `stream` object provided as an argument. Each of these values should be printed in a separate line.

Whenever the function is called without the `stream` argument, it should use an instance of `EvenStream` class defined in the code stubs below as the value of `stream`.

Your function will be tested on several cases by the locked template code.

Input Format

The input is read by the provided locked code template. In the first line, there is a single integer q denoting the number of queries. Each of the following q lines contains a `stream_name` followed by integer n, and it corresponds to a single test for your function.

Constraints

- $1 \leq q \leq 100$
- $1 \leq n \leq 10$

Output Format

The output is produced by the provided and locked code template. For each of the queries `(stream_name, n)`, if the `stream_name` is `even` then `print_from_stream(n)` is called. Otherwise, if the `stream_name` is `odd`, then `print_from_stream(n, OddStream())` is called.

Sample Input 0

```
3
odd 2
even 3
odd 5
```

Sample Output 0

```
1
3
0
2
4
1
3
5
7
9
```

Explanation 0

There are 3 queries in the sample.

In the first query, the function `print_from_stream(2, OddStream())` is exectuted, which leads to printing values 1 and 3 in separated lines as the first two non-negative odd numbers.

In the second query, the function `print_from_stream(3)` is exectuted, which leads to printing values $2, 4$ and 6 in separated lines as the first three non-negative even numbers.

In the third query, the function `print_from_stream(5, OddStream())` is exectuted, which leads to printing values $1, 3, 5, 7$ and 9 in separated lines as the first five non-negative odd numbers.