Problem D: Task

Source: `task.{c,cpp,java}`
Input: `console {stdin,cin,System.in}`
Output: `console {stdout,cout,System.out}`

In most recipes, certain tasks have to be done before others. For each task, if we are given a list of other tasks that it depends on, then it is relatively straightforward to come up with a schedule of tasks that satisfies the dependencies and produces a stunning dish. Many of us know that this can be solved by some algorithm called toplogical sort.

But life is not so easy sometimes. For example, here is a recipe for making pizza dough:

1. Mix the yeast with warm water, wait for 5 to 10 minutes.
2. Mix the the remaining ingredients 7 to 9 minutes.
3. Mix the yeast and the remaining ingredients together for 10 to 15 minutes.
4. Wait 90 to 120 minutes for the dough to rise.
5. Punch the dough and let it rest for 10 to 15 minutes.
6. Roll the dough.

In this case, tasks 1 and 2 may be scheduled after the first minute (we always spend the first minute to read the recipe and come up with a plan). The earliest task 3 may be started is at 8 minutes, and task 4 may start at 18 minutes after the start, and so on. This recipe is relatively simple, but if some tasks have many dependent tasks then scheduling can become unmanageable. Sometimes, the recipe may in fact be impossible to execute. For example, consider the following abstract recipe:

1. task 1
2. after task 1 but within 2 minutes of it, do task 2
3. at least 3 minutes after task 2 but within 2 minutes of task 1, do task 3

In this problem, you are given a number of tasks. Some tasks are related to another based on their starting times. You are asked to assign a starting time to each task to satisfy all constraints if possible, or report that no valid schedule is possible.

Input

The input consists of a number of cases. The first line of each case gives the number of tasks **n**, (**1 ≤ n ≤ 100**). This is followed by a line containing a non-negative integer **m** giving the number of constraints. Each of the next **m** lines specify a constraint. The two possible forms of constraints are:

```
task i starts at least A minutes later than task j
task i starts within A minutes of the starting time of task j
```

where **i** and **j** are the task numbers of two different tasks (**1 ≤ i, j ≤ n**), and **A** is a non-negative integer (**A ≤ 150**). The first form states that task **i** must start at least **A** minutes later than the start time of task **j**. The second form states that task **i** must start no earlier than task **j**, and within **A** minutes of the starting time of task **j**. There may be multiple constraints involving the same pair of tasks. Note that **at least** and **within** include the end points (i.e. if task 1 starts at 1 minute and task 2 starts at 4 minutes, then task 2 starts at least 3 minutes later than task 1, and within 3 minutes of the starting time of task 1).

The input is terminated by **n = 0**.

Output

For each case, output a single line containing the starting times of task 1 through task n separated by a single space. Each starting time should specify the minute at which the task starts. The starting time of each task should be positive and less than 1000000. There may be many possible schedules, and any valid schedule will be accepted. If no valid schedule is possible, print **Impossible.** on a line instead.

Sample input

```
6
10
task 3 starts at least 5 minutes later than task 1
task 3 starts within 10 minutes of the starting time of task 1
task 3 starts at least 7 minutes later than task 2
task 3 starts within 9 minutes of the starting time of task 2
task 4 starts at least 10 minutes later than task 3
task 4 starts within 15 minutes of the starting time of task 3
task 5 starts at least 90 minutes later than task 4
task 5 starts within 120 minutes of the starting time of task 4
task 6 starts at least 10 minutes later than task 5
task 6 starts within 15 minutes of the starting time of task 5
3
4
task 2 starts at least 0 minutes later than task 1
task 2 starts within 2 minutes of the starting time of task 1
task 3 starts at least 3 minutes later than task 2
task 3 starts within 2 minutes of the starting time of task 1
0
```

Sample Output

```
3 1 8 18 108 118
Impossible.
```