Samsung mySingle

Job Schedule

You have V jobs to complete. A set of prerequisite relations are enforced with the jobs;

some job can start only after some other job(s) has(have) been completed.

You are given a graph to represent these relations.

In this graph, each job corresponds to a vertex and a prerequisite relation corresponds to a directed edge.

There can be a cycle in this graph. (A cycle is a path starting from a vertex and returning to the same vertex.) The following is an example graph.

In this graph, job 1 can start only after job 4. Job 6 can start only after jobs 5 and 7 both have been completed. You can see that there is no cycle in the graph.

You have to complete all the jobs one by one not violating the prerequisite relations. A possible sequence with respect to the above graph is

8, 9, 4, 1, 5, 2, 3, 7, 6.

Another sequence is also valid:
4, 1, 2, 3, 8, 5, 7, 6, 9.

It is usual that there exists more than one valid sequence for a given graph. The following is an invalid sequence:

4, 1, 5, 2, 3, 7, 6, 8, 9.

Job 5 precedes job 8 in this sequence; in the prerequisite graph, however, job 5 can start only after job 8 has been completed; thus, the sequence is not possible.

Given V jobs and a set of prerequisite relations, generate a program that finds a valid sequence of jobs. When there is more than one valid sequence, you only have to provide one of them. Since it is not possible to have a valid sequence for a graph with a cycle, your program has to detect such graphs. In case of a graph with no cycle, there exists at least a valid sequence.

[Constraints]
The number of jobs: 3<=V<=1000.
Time limit: 1 second in total for 10 test cases.

[Input]
10 test cases are given. Each case occupies two lines; thus there are 20 lines in total. For each case, the first line has V and E, the numbers of jobs and edges, respectively. The next line enumerates E (directed) edges. An edge is represented by two jobs. For example, an edge from job 5 to job 28 is represented by 5 28. The indices of jobs are 1 through V. Each number in a line is separated by a space.

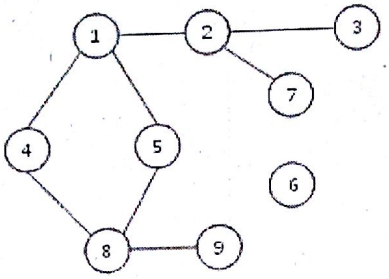
[Output]
You print 10 answers in 10 lines. Each line starts with #x where x means the index of a test case. Then, it puts a space and enumerates a job sequence in the same line. Each member in the line is separated by a space. If the graph has a cycle, you just print 0.

[I/O Example]
Input (20 lines in total)
9 9
4 1 1 2 2 3 2 7 5 6 7 6 1 5 8 5 8 9
5 4 ← Starting the case 1
1 2 2 3 4 1 1 5
5 5 ← Starting the case 2
1 2 2 3 3 1 4 1 1 5
...
← Starting the case 3

Output (10 lines in total)
#1 8 9 4 1 5 2 3 7 6
#2 4 1 2 3 5
#3 0

Bipartitioning Cities

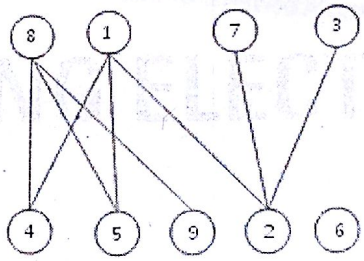
You are given a network of roads between N cities. The following is an example. Here a vertex means a city and an edge means a bidirectional road between a pair of cities.



You have to partition the cities into two mutually exclusive groups, satisfying the following condition:

There is no road connecting two cities in the same group.

The following is an example partitioning with the above graph. The cities are partitioned into those in the top level and those in the bottom level. You can see that there is no road between cities in the same group. Here, the city 6 can belong to any of the two groups.



When you are given a road network with N cities, generate a program that partitions the cities into two groups satisfying the condition described above. If you can not partition them appropriately, you have to specify it.

[Constraints]

The number of cities: $5 \le N \le 1000$.

The number of edges: $1 \le E$.

Time limit: 1 second in total for the 10 test cases.

[Input]

You are given 10 test cases. A test case has two lines; thus there are 20 lines in total. In each test case, the first line has N (the number of cities) and E (the number of roads) separated by a space. The next line enumerates E roads. A road consists of the two cities it connects. For example, the road connecting cities 5 and 28 is represented by "5 28" or "28 5". The indices of cities are 1 through N. All adjacent numbers in a line are each separated by a space.

[Output]

Output the 10 answers in 10 lines. Each line starts with '#x', where x means the index of a test case, and puts a space, and prints the answer. The answer enumerates only one of the two groups. The answer starts with K, the number of cities in the group, and then enumerates the K cities. If it is not possible to properly partition, print –1. If there is more than one appropriate partitioning, answer just one of them. All numbers in a line are each separated by a space.

[I/O Example]

Input (20 lines in total)

```
9 8                                   ← Starting test case 1
4 1 1 2 2 3 7 2 1 5 8 4 5 8 8 9
7 10                                  ← Starting test case 2
1 2 2 3 1 6 4 6 2 4 2 7 2 5 6 7 3 5 5 7
...
```

Output (10 lines in total)

```
#1 4 8 1 7 3
#2 -1
...
```

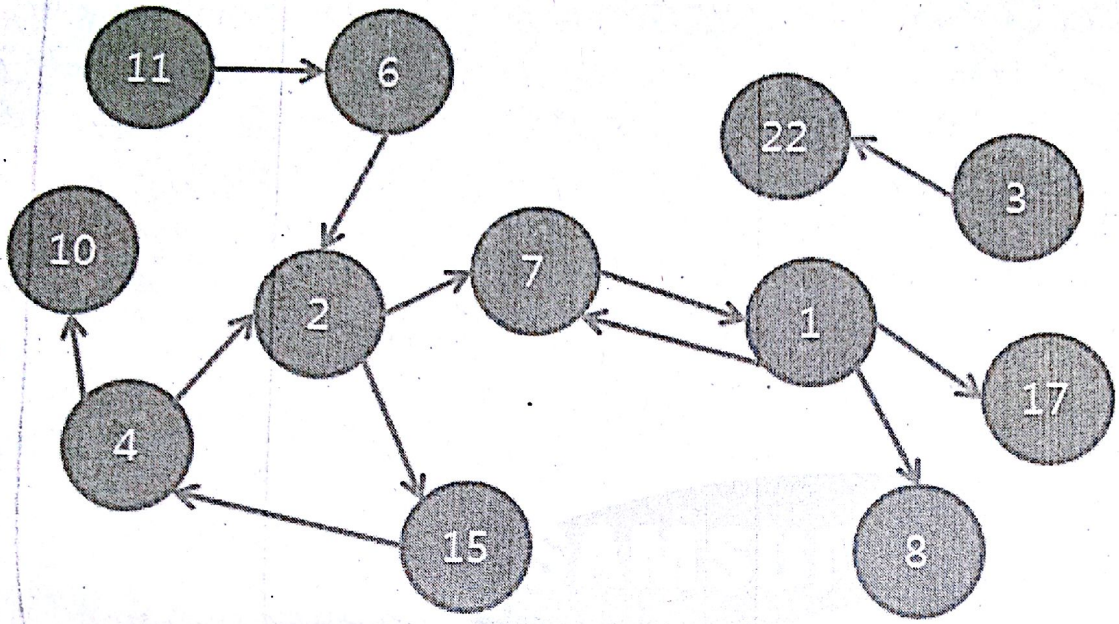
(You may output "#1 5 4 5 9 2 6" instead in the first case.)

[Problem] Write a C function which meets following conditions. (Development time: 4 hours)

Given information about a 'network of emergency contacts' and a person who is on duty of making calls first, write a C function that finds the person who has the largest number among those who are contacted last.

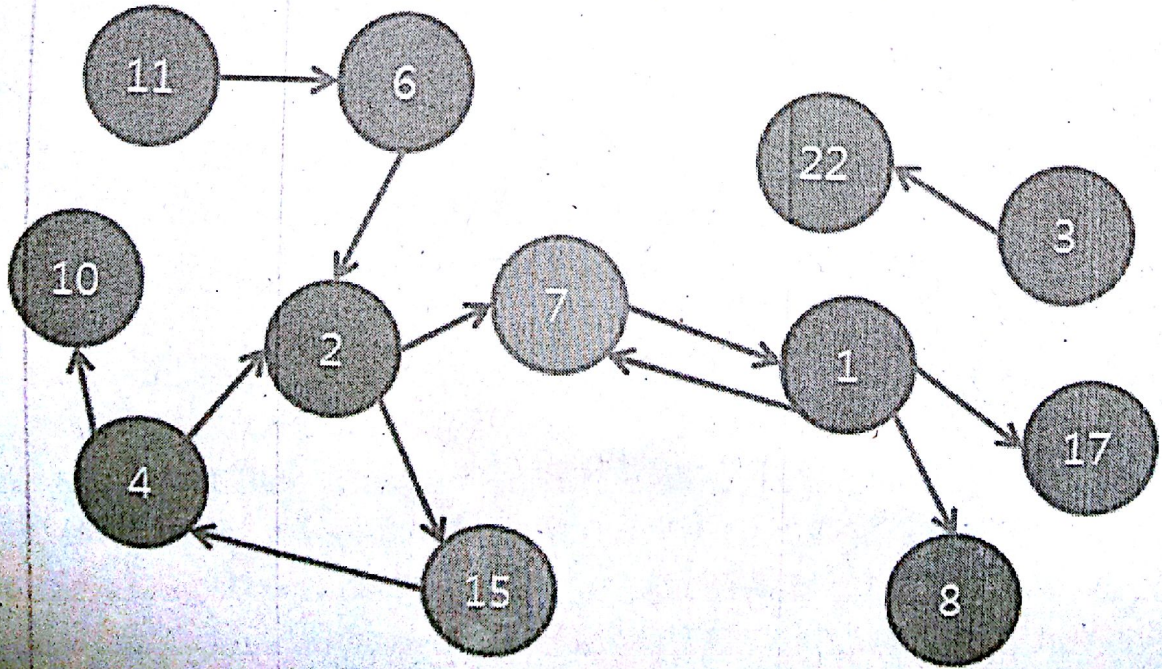
[Example]

Below is the picture showing a 'network of emergency contacts.'

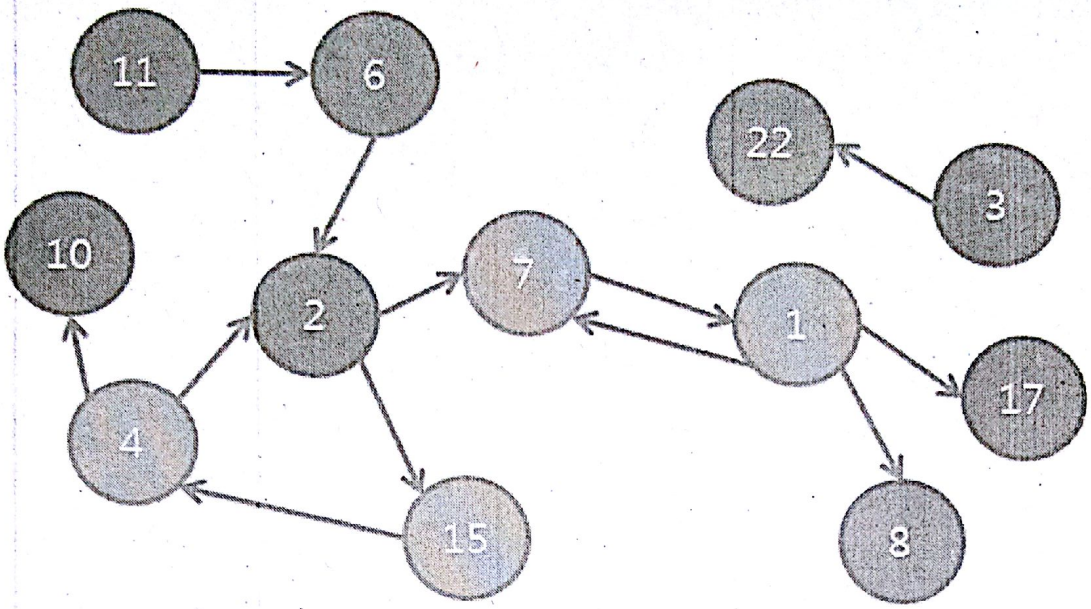


Each circle means each individual person and the number in it presents the number of each person. A red circle indicates the person who is on duty of making calls first. Arrows mean the directions in which contacting is possible. In this example above, the number 7 and 1 can contact each other, however, in case of the number 2 and 7, the number 2 can contact the number 7, but not vise versa.

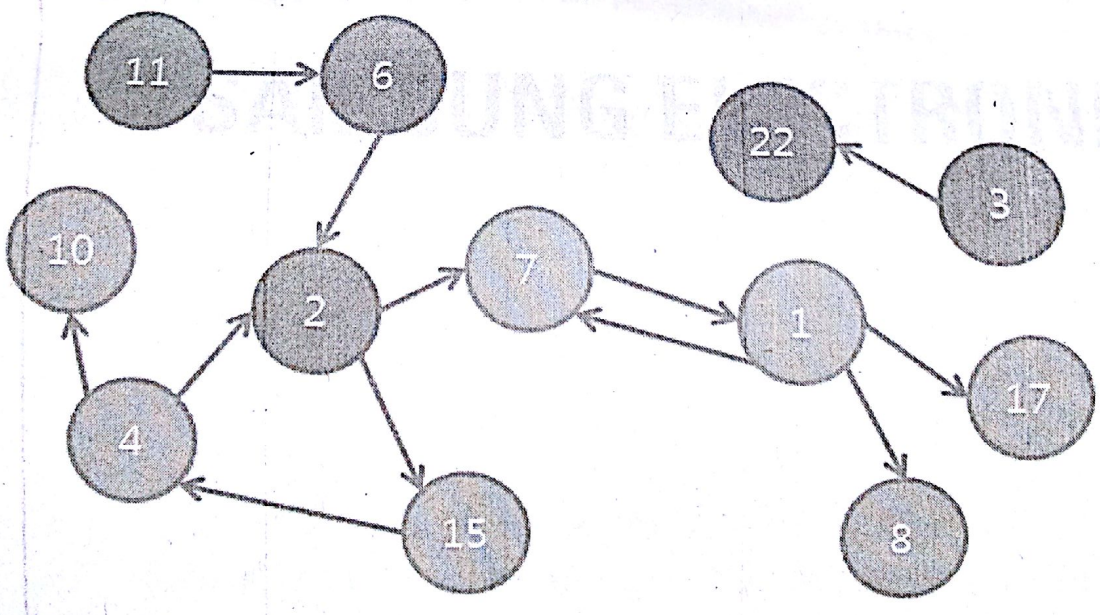
If contacting is started through this 'network of emergency contacts,' the number 2 who is on duty of making calls first will contact the number 7 and 15 simultaneously. (Assume that a conference call is used.)



Then, the number 7 contacts the number 1 and the number 15 contacts the number 4. (Assume that contacting happens at the same time.)



And then, the number 1 simultaneously contacts the number 8 and 17 and at the same time, the number 4 contacts the number 10 like below. Since the number 7 and 2 have already been contacted, they will not be contacted again.



The picture above shows the final state in which all possible contacts are reached. There are three persons, the number 8, 10, and 17, who are contacted last. Among those, the largest number is 17, therefore, return 17.

※ The number 3, 6, 11, 22 will not be able to get reached after all.

[Constraints]

Any external libraries or the header file cannot be used. (You will be failed if you use #include <stdio.h>,etc)

When evaluating your solution, main.cpp will not be changed.

The number of contact persons is maximum 100, and the number which can be assigned is from 1 to 100, inclusive. But, as the number 5 does not exist in this example, there can be the numbers which are vacant.
If one person is able to reach many people, all contacts are done at the same time though a conference call.
The speed at which contact is reached is always constant. (The speed at which a person who received a call makes a call to the next person is the same.)
Information about a network of emergency contacts is shared in advance and the person who was contacted once will not be contacted again. Like the number 3,6,11, and 22 in this example, there can be the persons who will not be able to get reached after all.

[A function that you have to implement]

```
int test_main(int data[300], int startNum)
```

data[300] : data about a network of emergency contacts is given.
Data is interpreted in order of{from, to, from, to, ...}
In the example above, the data is like {2, 7, 11, 6, 6, 2, 2, 15, 15, 4, 4, 2, 4, 10, 7, 1, 1, 7, 1, 8, 1, 17, 3, 22}.
(Pay attention to the fact that there are 24 numbers- just for this example- not 300.)
However, the place of these pairs does not make any difference, therefore, following input indicates the same 'network of emergency contacts'. (There can be various inputs which present the same 'network of emergency contacts.')
{1, 17, 3, 22, 1, 8, 1, 7, 7, 1, 2, 7, 2, 15, 15, 4, 6, 2, 11, 6, 4, 10, 4, 2}

There can be a case that the same {from, to} pair is repeated several times as follows, and there is no difference between a case in which the same pair is recorded once and recorded more times.
{**1, 17, 1, 17, 1, 17**, 3, 22, 1, 8, 1, 7, 7, 1, 2, 7, 2, 15, 15, 4, 6, 2, **11, 6,** 4, 10, **11, 6**, 4, 2}

startNum : information about a person who is on duty of contacting first (In this example, it is 2)

return : return the number of the person who has the largest number among those who are contacted last.

```
[file 1  - your_ID_number.cpp]
// DO NOT INCLUDE ANY FILES
int test_main(int data[300], int startNum)
{
    return 0;
}
```

```
[file 2  - main.cpp]
#include <stdio.h>
#include <stdlib.h>

int test_main(int data[300], int startNum);
```

```
static void build_data(int data[300])
{
    for (int i = 0; i < 300; i++)
    {
        data[i] = rand() % 100 + 1;
    }
}

void main(void)
{
    int data[300];

    for (int l = 0; l < 10; l++)
    {
        build_data(data);
        printf("%d\n", test_main(data, data[0]));
    }
}
```

Counting 99

You are given a sequence of integers. You have to count the number of 99s in the sequence. For example, given the following sequence,

25, 2, 32, 99, 1, 1024, 555, 35, 99, 99, 2, 9, -2, 87, -99

99 appears three times; so the answer is 3. If there is no 99 in the sequence, the answer is naturally 0.

Given a sequence of 100 integers, generate a program that counts the number of 99s.

[Constraints]

Time limit: 0.1 seconds.

[Input]

100 integers are enumerated in one line each separated by a space. Each integer is not smaller than –10,000 and not greater than 10,000.

[Output]

The number of 99s.

[I/O Example 1]

Input

(Here, just 20 integers due to space limit. Test case will contain 100 integers.)

```
25 2 32 99 1 1024 555 35 99 99 2 9 -2 87 -99 3749 7109 -6124 -6459 620
```

Output

```
3
```

[I/O Example 2]

Input

(Here, just 10 integers due to space limit. Test case will contain 100 integers.)

```
-8997 9940 -999 999 -99 990 9999 333 -903 -3013
```

Output

```
0
```

Blackjack

You are given a sequence of N positive integers. Each integer belongs to 1, 2, ..., 13 and the same integer may appear more than once. You have to keep adding the N integers one by one in the given order. The sum must not exceed 21. What is the maximum sum you can get?

For example, you are given four integers <5, 5, 6, 11>. The possible sums are 5, 5+5=10, 5+5+6=16, and 5+5+6+11=27; these are all you can get. The maximum among them not exceeding 21 is 16. If you are given three integers <8, 2, 5>, the maximum sum is 15.

There is another rule. The integer 1 can be counted either as 1 or as 11; it has some flexibility. For example, if you are given three integers <8, 1, 3>, the possible sums are 8, 9, 12 or 8, 19, 22. The maximum not exceeding 21 is 19.

when you are given a sequence of N integers from 1, 2, ..., 13 and keep adding them in the given order, under the rule described above, generate a program that finds the maximum sum not exceeing 21.

[Constraints]

The number of integers: $1 \le N \le 10$.

Time limit: 10 seconds in total for the 10 test cases.

[Input]

You are given 10 test cases. A test case has one line. Each line starts with N (the number of integers) and then enumerates a sequence of N integers, each separated by a space.

[Output]

Output the 10 answers in 10 lines. Each line looks like '#x answer' where x is the index of a test case. '#x' and 'answer' are separated by a space.

[I/O Example]

Input (10 lines in total)

```
4 5 5 6 11
3 8 2 5
3 8 1 3
...
```

Output (10 lines in total)

```
#1 16
#2 15
#3 19
```

[Problem] Please write a C or Java program according to the following conditions

There are two jugs named A and B with limited defined capacities to hold water. There are no markings on a jug and it may be of an irregular shape. If a jug has 5 litre capacity, you know for sure only that, if you fill it, you will get 5 litre in the jug. Moreover, you can only logically deduce a certain quantity of water in a jug after few predefined jug operations. For example, let us think you have a 5 litre jug and a 3 litre jug. Now, to obtain 2 litre water you may want to do something like – fill 5 litre jug, transfer water into 3 litre jug . Now, you can safely infer that there is 2 litre of water in the 5 litre jug. A number of such operations can effectively help you obtain required amount of water in one of the jugs.

Now, the problem is, if you are given a pair of jugs with known capacities and an unlimited supply of water, can you deduce a required amount of water through a series of logical operations such as – fill, empty & move. Provide a specified amount of water using filling, moving and emptying a pair of jugs with marked capacities and un-limited supply of water.

Example – Given 5 and 3 litre jugs, if you have to extract 4 litres of water, your sequence of operations are as follows

Input : 5,3,4

A will correspond to jug with 5 litre capacity and
B to that with 3 litre respectively.

• Fill A	fill(a);	// Now A contains 5L, B 0L
• Move A to B	move(a,b);	// Now A contains 2L, B 3L
• Empty B	empty(b);	// Now A contains 2L, B 0L
• Move A to B	move(a,b);	// Now A contains 0L, B 2L
• Fill A	fill(a);	// Now A contains 5L, B 2L
• Move A to B	move(a,b);	// Now A contains 4L, B 3L
• Output A (it means that you output A as the jug that has specified amount of water which is 4 litre)	output(a);	// Verify whether A contains required water

Evaluation rules

- Output jug must contain requested output water quantity in order to PASS for junior level for all iterations
- Number of jug operations should be minimum for senior level
- For a given pair of jugs a specified output may not be theoretically possible. You should find such cases and output -1 with zero moves for expert level along with minimum operations where it is possible

Note: For Junior and senior levels you can safely assume requested output quantity is always possible.

We evaluate for expert only if you reach senior level and in this case one should not assume requested output quantity is possible.