PROMPT: In a physical education class, for performing a rigorous endurance exercise, the instructor of the class tries to divide the students into two groups. The instructor has already compiled a list of students in class, and noted down the interactions among students. Specifically, she has created a social network where each node in the network corresponds to a student, and an edge between two students corresponds to friendships between the students. In order to improve camaraderie, she decides to break the teams in such a way that no friend of a student are in the same team. That is, if student x is picked to be in team 1, then all of x's friends will be in team 2, and their friends will be in team 1, and so on. Of course, if the network has a three cycle (a cycle of three friends, or any odd cycle), then such a division is impossible.

Please write a program, that given a social network, informs whether it is possible to divide the students into two teams as above. If so, please print for each student, which team they will belong to.

Input and Output:
The first input line has two integers n and m, where n is the number of students, and m is the number of friendships between students. The students are numbered 1, 2, . . . , n. Then, there are m lines describing the friendships. Each line has two integers a and b, where students a and b are friends. Every friendship is between two different students. You can assume that there is at most one friendship between any two students.
As output, please print an example of how to build the teams. For each student, print "1" or "2" depending on to which team the student will be assigned. You can print any valid team assignment. That is, your program will be judged as correct as long as some individual x is on some team (say Team 1), and all of their friends are in the other team (say Team 2). But note that x could also be on Team 2, while their friends are in the other team. So as long as your output is consistent with the specifications, your program will be judged as correct.
3

Please print the team assignments on a single line, all elements space separated. If there are no solutions, print "impossible". A sample input and output is shown below, and is also in files forming-teams.in1 and forming-teams.out1 respectively.
Sample Input (see forming-teams.in1):

```
10 13
1 2
1 4
2 3
3 5
3 6
3 7
1 5
1 7
2 10
```

4 10
4 8
4 9
9 2
Sample Output (see forming-teams.out1):
1 2 1 2 2 2 2 1 1 1

Some startup code is provided in FormingTeams.java. This program reads the input file and stores the network in an adjacency list representation, which is an array of a list of neighbors for each node. Nodes are labeled 1 to n. We use ArrayList objects to store the neighbors for each node, as this already gives us an expanding array. So the object adjacency list[x] stores the neighbors of node x. You may use the Java documentation for ArrayList, at https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html.
To iterate through the adjacency list and print its neighbors, we can use a for loop like;

```
for (int x: adjacency_list) {
for (Integer y: adjacency_list[x]) { // this loops through the neighbor of x
int z = y.intValue(); // converts the Integer object to int
// z a neighbor of x.
...
}
}
```

Your program should work for $n \leq 105$ and $m \leq 2 \cdot 105$
. As a hint, modify
depth first search to assign nodes as either in team 1 or 2. If you ever assign a node as both team 1 and 2, you will know that there is an odd cycle, and it is impossible to form teams.