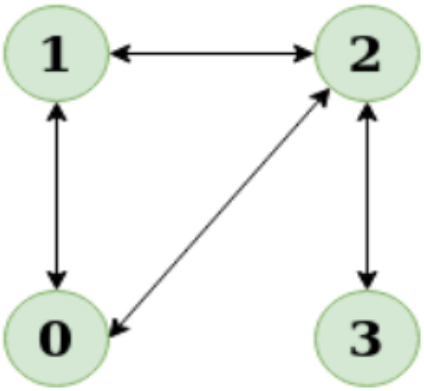
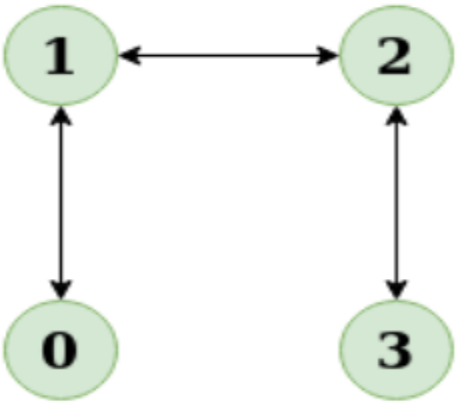
- Assignment Name: Lab7
- Description: Question based on graph traversal (DFS)[Cycle detection]
- Total Marks: 10
- Problem Title**: Detect a cycle in the given undirected graph in the form of an edge list**
- Marks allotted: 10
- Description:

Using the given edge list which contains all the edges in an undirected graph and traverses through the graph to detect any cycles.

The cycle detection algorithm can be done using either DFS or BFS as per the student's convenience and must simply return true if a cycle is detected and false if there is no cycle that is detected.



The above-given graph must return True.



The above-given graph must return False.

You are required to implement the following graph operation as a part of this lab exercise:

```
unsigned int cycle_finder(const edge *edges , unsigned int n, unsigned int order)
{

}
```

This function has 3 arguments which are the edge list, n which is the number of EDGES in the graph and order which is the number of VERTICES in the graph.

You may call any helper functions that you have defined from cycle finder in order to solve the problem and are expected to return 1 if the is a cycle found and 0 if there is no cycle detected.

** do note that unsigned int has been used everywhere to signify the use of non-negative values for the vertices only.
You are supposed to use the code provided to you via the boilerplate code.
The skeleton of the edgelist (aka graph) is provided to you via “struct” so you need not code that.

```
/*edge list structure for each edge */
typedef struct{
    unsigned int first;
    unsigned int second;
}edge;
```

Here first signifies the first vertex of an edge and the second signifies the second vertex of an edge.

Functions to initialise the edge list, and free edge list has already been implemented in the boilerplate code.
USING THE BOILERPLATE CODE PROVIDED IS MANDATORY.

- Input format:
  There are 2 + n lines in the input format
  The first line which takes the number of edges in the edge list (n)
  The second line takes the order (the number of vertices in the graph) (order)
  The next n lines take the edges (each of the vertices space separated)
- Output format:
  The output must simply return 1 or 0 for true or false if a cycle is detected respectively.

- Sample test cases:

1. TC #1:
    - Input:
      3
      3
      0 1
      1 2
      2 0

    - Expected output:
      cyclic

2. TC #2:
    - Input:
      2
      2
      0 1
      1 2

    - Output:
      acyclic