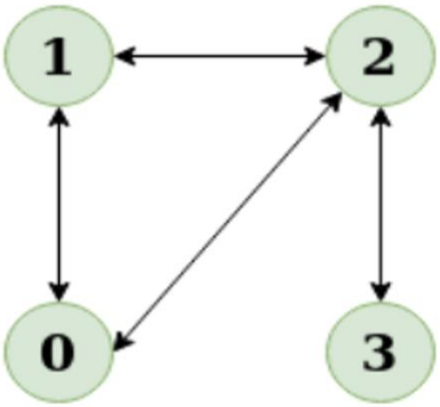
Assignment Name: Lab7

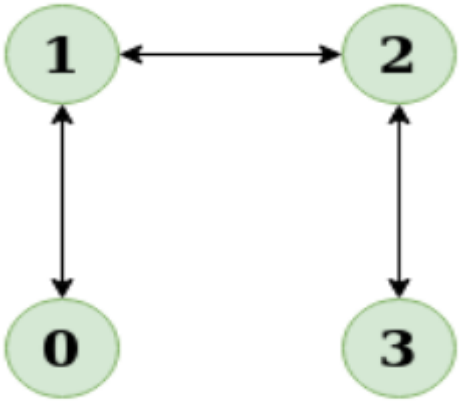
- Description: Question based on graph traversal (DFS)[Cycle detection]
- Total Marks: 10
- Deadline: Depends on respective lab hrs
- Problem Title**: Detect a cycle in the given undirected graph in the form of an edge list**
- Marks allotted: 10
- Description:

This week's lab exercise is to detect if a given undirected graph is cyclic or acyclic.
A graph is called cyclic if it has at least one cycle.
And an acyclic graph has no cycles at all.

The cycle detection algorithm can be coded either using DFS or BFS as per student's convenience.
You are required to complete the 'cycle_finder' function which returns true if the given graph has cycle, and false if it is acyclic.



The above graph must return True.



The above 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 parameters namely, the edge list ‘edges’, the number of edges in the graph ‘n’, and the number of vertices in the graph ‘order’.

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 there is a cycle found and 0 if there is no cycle detected.

NOTE:
1. unsigned int has been used everywhere to signify the use of non-negative values for the vertices only.
2. You are supposed to use the code provided to you via the boilerplate code.
3. The skeleton of the edge list (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:
  You are not required to print anything as a part of this lab exercise. The function must return 1 if there is a cycle and 0 if the graph is acyclic.

- Sample test cases:
  - #TC 1:
    - Input:
      3
      3
      0 1
      1 2
      2 0
    - Output:
      Cyclic

  - #TC 2:
    - Input:
      2
      3
      0 1
      1 2
    - Output:
      Acyclic