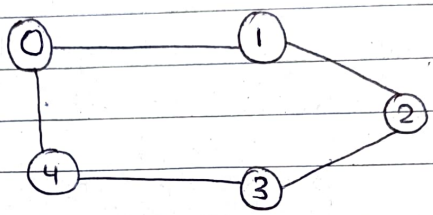
Date _____ Page _____

Day-200

Graph-4

* Cycle Detection:



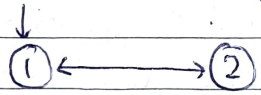
⇉ Here, we have to detect cycle in the graph.

⇉ For that, First, we will use DFS.

⇉ So, we will stand from a node & if we get back to that node, we detect a cycle.

⇉ But it is not always true.

Ex:



⇉ So, we will start from ① & than visit ② and than we again visit ①. But this not a cycle.

⇉ So, we don't have to take previous node or skip the previous node.

⇉ So, if we visit a visited node and that node is not the parent node then this is the cycle.

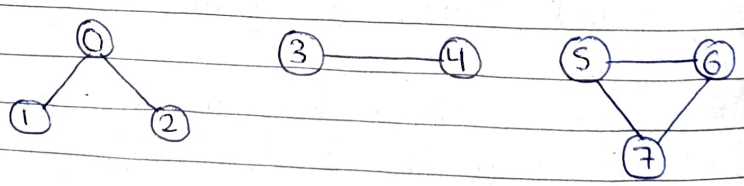
Date _____ Page _____

Code

```
bool detectCycle (int node , int parent ,
    vector <int> adj[ ] , vector <bool> &visited ){
    visited [node ] = 1 ;
    for( int j =0 ; j < adj[node ]. ~~adj~~ size() ; j++){
        if( parent == adj [node ][ j ] )
            continue ;
        if( visited [adj [node ][j ] ] == 1 )
            return 1 ;
        if(detect Cycle (adj [node][ j ] , node ,
            adj , visi ))
            return 1 ;
    }
    return 0 ;
}
```

⇉ There is an error in our code.

⇉ Suppose all the nodes of the graphs are not connected.

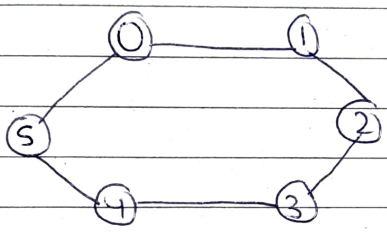


⇉ we only visited 0, 1, 2 but not visited ③, ④, ⑤, ⑥ & ⑦.

Date _____ Page _____

⇒ So, we will visit other nodes with the help of visited array.

⇒ By using BFS



⇒ So, when we are using BFS, if we visited any node more than one time then the cycle present.

⇒ So, we will use queue and store two info. — node and it's parent node.

⇒ we will check all the previous conditions that we check in the previous approach.