Implementation of the graph using vector

1. `vector<vector<int> printAdjacency( int n, int m, vector<vector<int>> & edges )`

what is vector<vector<int>>

It is a kind of 2 D array, and its element can be accessed by arr[i][j], where i is the row and j is the column

array = { {1,2,3} , {4,5}, {6,7,8,9} }

1	2	3	0
4	5	0	0
6	7	8	9

array = { {1,2,3},
{4,5},
{6,7,8,9}
}

Integers get filled, row by row. Take care that initial size is defined to incorporate maximum row, column entry and the entries I do not pass get replaced by 0.

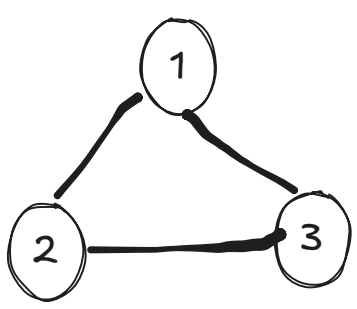
m is the number of edges and n is the number of nodes.

edges is also a 2D Vector and it contains m rows and 2 columns, we can also access the edges by
u = edges[i][0]
v = edges[i][1]

i	0	1
	1	2
	2	3
	3	1

2. `vector<int> ans[n]`

Our strategy is to create an array called as ans which has n vectors, each each index denotes the node, so I will just add elements in the vector present at the indexes which are adjacent and now I will get the the vector called as ans in which each index is a node and at each node there is a vector containing all the elements adjacent to it.



ans = { { } , { 2,3}, {1,3} , {1,2} }

3. `vector<vector<int>> adj(n)`

This also essentially contains n indexes and in this we just put the values in a certain manner,

```
for( int i = 0; i < n; i ++ ){
    adj[i].push_back(i);
```

adj becomes { {0} {1} {2} {3} }

```
for( int i = 0; i < ans[i].size; i++){
    adj[i].push_back(ans[i][j]);
```

adj becomes { {0} , { 1 2 3}, { 2 1 3 }, {3 1 2} }