```
class Node(object):
    def __init__(self, name):
        self.name = name
        self.neighbours = []

def set_nodes_of_edge(graph, node_name, node_list, node):
    if node_name not in node_list:
        node_list.append(node_name)
        node.append(Node(node_name))
        graph.append(node[0])
    else:
        for found_node in graph:
            if found_node.name == node_name:
                node.append(found_node)
                break

def load_graph_edges(edge_list):
    node_list = []
    graph = []
    for edge in edge_list:
        node_a_name = edge[0]
        node_b_name = edge[1]
        node_a = []
        node_b = []

        set_nodes_of_edge(graph, node_a_name, node_list, node_a)
        set_nodes_of_edge(graph, node_b_name, node_list, node_b)

        # Add nodeA/B to each other as neighbours
        for node in graph:
            if node.name == node_a_name:
                node.neighbours.append(node_b[0])
            if node.name == node_b_name:
                node.neighbours.append(node_a[0])

    return graph

def get_bfs_list(edges):
    #Load edges
    graph = load_graph_edges(edges)

    #BFS
    visit_queue = [graph[0]]    # Queue of nodes to visit (and add neighbours at
 front of queue to back)
    visited = []                # Visited nodes to exclude from queuing to visit
_queue
    bfs_list = []               # BFS ordering

    # Read visit_queue until all nodes have been exhausted
    while len(visit_queue) > 0:
        bfs_list.append(visit_queue[0])
        visited.append(visit_queue[0].name)

        # Append all neighbours of front of visit queue if they haven't been vis
ited yet
        for node in visit_queue[0].neighbours:
            if not node.name in visited:
                visit_queue.append(node)
                visited.append(node.name)
        visit_queue.pop(0)

    return bfs_list

#edges = [['A','B'], ['A','C'], ['B','C'], ['B','D'], ['B','E'], ['C','D'], ['C'
,'E'], ['D','E'],['D','F'],['E','F']]
#edges = [['A','D'], ['A','E'], ['D','C'], ['D','B'], ['E','B'], ['B','C']]
```