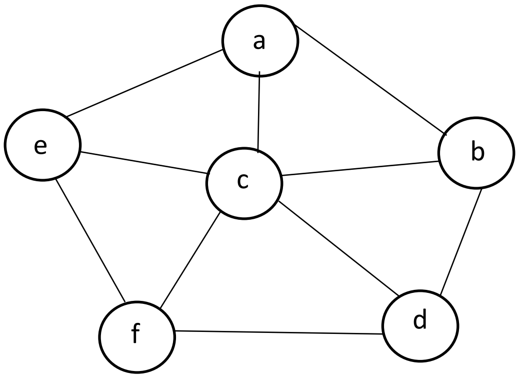
COLOURING PROBLEM

<u>Problem interpretation</u>: The problem is interpreted to be a graph colouring problem. Each small partition in the problem statement is considered to be a node. Each such partition shares its sides with some other partitions which are referred to as its neighbours. No node can have same colour as its neighbour.



<u>Input</u> : An object named "probStmt" with the name of nodes as keys and object of properties as value. This value object in turn contains name, neighbours and colour properties for that node.

<u>Output</u> : The program displays the name of each node and the colour assigned to it in the console. (However, if needed the colours can also be accessed in the "probStmt" object since the program changes those value in the probStmt object itself)

<u>Program Flow:</u>

Program flow for initialization and BFS:

1. The graph is hardcoded as an object
2. Choose any node as root node.
3. An empty queue is instantiated to hold unvisited nodes of the graph, add root node to this queue.
4. Assign a colour to the root node.
5. Repeat steps 6 to 8 till every node is visited
6. Pop the first node from the unvisited queue and store the value of its neighbours in a variable
7. Repeat step 8 till every neighbour is visited
8. If the current neighbour is not assigned a colour, assign it a colour, push the current neighbour into the unvisited queue. Else, go back to step 7
9. Display the result in the form of each node and its colour

Program flow for colour assignment function:

1. Store the available colours in a local colour queue and neighbours of current node in a variable
2. Continue steps 3 to 4 till all the neighbours of current node are iterated over
3. Store the value of colour for the current neighbour in a variable
4. If the neighbour is coloured, remove that colour from the local colour queue. Else, go back to step 2
5. Pop a colour from the local colour queue and colour the node with that colour

<u>Possible improvements:</u>

1. Dynamic creation of nodes instead of hardcoding
2. Handling error caused by a disconnected graph
3. Handling error caused by insufficient colours provided for colouring the graph