In [2]:
```python
def dfs(graph, node, visited):
    if node not in visited:
        print(node)
        visited.add(node)
        for neighbor in graph[node]:
            dfs(graph, neighbor, visited)

# sample graph represented as an adjacency list
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F','G'],
    'D': ['B'],
    'E': ['B'],
    'F': ['C'],
    'G':['C']
}
dfs(graph, 'A', set())
```

```
A
B
D
E
C
F
G
```

In [3]:
```python
def bfs(graph, start):
    visited, queue = set(), [start]
    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            visited.add(vertex)
            queue.extend(graph[vertex]-visited)
    return visited

# sample graph represented as an adjacency list
graph = {
 'A': set(['B', 'C']),
 'B': set(['A', 'D', 'E']),
 'C': set(['A', 'F','G']),
 'D': set(['B']),
 'E': set(['B']),
 'F': set(['C']),
 'G': set(['C'])
}
bfs(graph, 'A')
```

Out[3]:
```
{'A', 'B', 'C', 'D', 'E', 'F', 'G'}
```

In [ ]: