In [75]:
```
gfs = {1:[2,5,8], 2:[3,4], 5:[6,7], 8:[7,9], 3:[], 4:[], 6:[], 7:[], 9:[]} #diction
ary
gfs
```

Out[75]:
```
{1: [2, 5, 8],
 2: [3, 4],
 5: [6, 7],
 8: [7, 9],
 3: [],
 4: [],
 6: [],
 7: [],
 9: []}
```

In [91]:
```
class mybfs:
    graph = {}
    queue = []
    explored = []
    def __init__ (self):
        print("my BFS")
    def push (self, node):
        self.queue.append(node)
    def pop (self):
        return self.queue.pop(0)
    def Graph (self, g):
        self.graph = g
    def showFrontier(self):
        print("Frontier: ", self.queue)
    def showExplored(self):
        print("Explored: ", self.explored)
    def showGraph (self):
        print(self.graph)
    def searching (self, start, goal):
        self.push(start)
        self.showFrontier()
        node = self.pop()
        while node!=goal:
            self.explored.append(node)
            self.showExplored()
            self.showFrontier()
            childs = self.graph[node]
            print(childs)
            for items in childs:
                if items not in self.explored:
                    if items not in self.queue:
                        self.push(items)
                        print(items)
            node = self.pop()
        if node==goal:
            print("Goal Found")
```

In [92]:
```
b = mybfs()
```

```
my BFS
```

In [93]:
```
b.Graph(gfs)
```

In [94]:
```
b.showGraph()
```

```
{1: [2, 5, 8], 2: [3, 4], 5: [6, 7], 8: [7, 9], 3: [], 4: [], 6: [], 7: [], 9:
[]}
```

In [95]:
```
b.searching(1,6)
```

```
Frontier:  [1]
Explored:  [1]
Frontier:  []
[2, 5, 8]
2
5
8
Explored:  [1, 2]
Frontier:  [5, 8]
[3, 4]
3
4
Explored:  [1, 2, 5]
Frontier:  [8, 3, 4]
[6, 7]
6
7
Explored:  [1, 2, 5, 8]
Frontier:  [3, 4, 6, 7]
[7, 9]
9
Explored:  [1, 2, 5, 8, 3]
Frontier:  [4, 6, 7, 9]
[]
Explored:  [1, 2, 5, 8, 3, 4]
Frontier:  [6, 7, 9]
[]
Goal Found
```

In [ ]:

In [ ]: