6.00 Handout, Lecture 22
(Not intended to make sense outside of lecture)

```
def shortestPath(graph, start, end, toPrint = False, visited = []):
    if toPrint:
        print start, end
    if not (graph.hasNode(start) and graph.hasNode(end)):
        raise ValueError('Start or end not in graph.')
    path = [str(start)]
    if start == end:
        return path
    shortest = None
    for node in graph.childrenOf(start):
        if (str(node) not in visited):
            visited = visited + [str(node)] #new list
            newPath = shortestPath(graph, node, end, toPrint, visited)
            if newPath == None:
                continue
            if (shortest == None or len(newPath) < len(shortest)):
                shortest = newPath
    if shortest != None:
        path = path + shortest
    else:
        path = None
    return path

def dpShortestPath(graph, start, end, visited = [], memo = {}):
    if not (graph.hasNode(start) and graph.hasNode(end)):
        raise ValueError('Start or end not in graph.')
    path = [str(start)]
    if start == end:
        return path
    shortest = None
    for node in graph.childrenOf(start):
        if (str(node) not in visited):
            visited = visited + [str(node)]
            try:
                newPath = memo[node, end]
            except:
                newPath = dpShortestPath(graph, node, end,
                                         visited, memo)
            if newPath == None:
                continue
            if (shortest == None or len(newPath) < len(shortest)):
                shortest = newPath
                memo[node, end] = newPath
    if shortest != None:
        path = path + shortest
    else:
        path = None
    return path
```

6.00SC Introduction to Computer Science and Programming
Spring 2011