Problem Wk.13.3.5: Obstacles [Optional]

Consider the following function for searching maps.

```
def mapTest(map, start, goal, searchFn = search.breadthFirstDP):
    actions = range(max([len(map[s]) for s in map]))

    def succFn(s, a):
        if a < len(map[s]):
            return map[s][a]
        else:
            return s

    def goalFn(s):
        return s == goal

    return searchFn(start, goalFn, actions, succFn)
```

We want to generalize map searching to allow us to specify that some actions and/or states are forbidden. For example, we could say that the road (arc) from 'A' to 'C' is not available and we could also say that state 'D' is forbidden. Attempting to perform a forbidden action or enter a forbidden states just leaves you in the current state. We will call the new function `mapObstTest` and it will have two new arguments:

- `badStates`: a list of state names that should never be visited.
- `badArcs`: a list of state name pairs, e.g. ('A', 'B'), describing state transitions that should never be attempted. This is directional, that is, it forbids going from 'A' and 'B' but not the other way.

Define `mapObstTest` below. It's a lot like `mapTest`; you should only need to modify the successor function. You can assume that your definition is being written in the `search.py` file, so you have access to all the definitions there.

If you do not know about the Python `in` operator, which checks whether an element is in a list, then read about it in the Python documentation.

You can debug using `tutor13Work.py`.

```
def mapObstTest(map, start, goal, badStates, badArcs,
                searchFn = breadthFirstDP):
    pass
```

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011