[15-112] Lecture 22

Lecture 22: Poll 1 (SOLO)
Which of the following correctly recursively, reverses a list?

A.
```
def reverseList(L, reversedSoFar=[]):
    if (L == [ ]):
        return reversedSoFar
    else:
        reversedSoFar.insert(0, L[0])
        return reverseList(L[1:], reversedSoFar)
```

B.
```
def reverseList(L):
    return reverseListHelper(L, [ ])

def reverseListHelper(L, reversedSoFar):
    if (L == [ ]):
        return reversedSoFar
    else:
        reversedSoFar.insert(0, L[0])
        return reverseListHelper(L[1:], reversedSoFar)
```

C. Neither
D. Both
E. I don't know

Lecture 22: Poll 1 (GROUP)
Which of the following correctly recursively, reverses a list?

A.
```
def reverseList(L, reversedSoFar=[]):
    if (L == [ ]):
        return reversedSoFar
    else:
        reversedSoFar.insert(0, L[0])
        return reverseList(L[1:], reversedSoFar)
```

B.
```
def reverseList(L):
    return reverseListHelper(L, [ ])

def reverseListHelper(L, reversedSoFar):
    if (L == [ ]):
        return reversedSoFar
    else:
        reversedSoFar.insert(0, L[0])
        return reverseListHelper(L[1:], reversedSoFar)
```

C. Neither
D. Both
E. I don't know

Backtracking pattern:

nQueens:

solve(board):

1. if all Qs on board
   return board solution!

2. for each action, if valid action:
   a. apply action
   b. recurse: result = solve(board)
   c. if result is success
      return result
   else
      undo action

3. return failure

Maze solver:

solve(maze, path, visited):

1. if all at goal
   return path as solution

2. for each action, if valid action:
   a. apply action
   b. recurse: result = solve(maze, path, visited)
   c. if result is success
      return result
   else
      undo action

3. return failure

Lecture 22: Poll 2

Which of the following is **not** a typical component of the backtracking algorithm?

A. Checking if the problem is solved in the base case
B. Looping through possible moves in the recursive case
C. Checking if a move is legal
D. Undoing a move if it does not lead to a solution
E. Returning a list of all valid solutions
F. None of the above
G. I don’t know

```
#######################################
# General Backtracking Template
#######################################

def generalBacktrackingAlgo(state, otherParams):
    # base case: check if we have reached a solution
    if isSolution(state):
        # we found a solution, so return it!
        return state
    else:
        # loop through all the possible moves from this state.
        for move in possibleMoves:
            # check if the move is legal
            if isLegal(move):
                # apply the move from this state
                newState = applyMove(state, move)
                # try to recursively solve from this new state
                solution = generalBacktrackingAlgo(newState, otherParams)
                if solution != None:
                    return solution
                # if we mutatingly applied the move to our state, undo move
                ''' undo move? '''
        # we looped through all the possible moves from this position, and
        # none of them worked out.
        return None
```