This is project work due at the end of the term. Because a non-trivial fraction of the class is behind schedule on the project, this assignment is shorter than most. :)

A. In qAlgorithms.py, implement the following function.

```
def grover(n, k, f):
    '''Assumes n >= 1, k >= 1. Assumes that k is small compared to 2^n.
    Implements the Grover core subroutine. The F parameter is an (n + 1)-qbit
    gate representing a function f : {0, 1}^n -> {0, 1} such that
    SUM_alpha f(alpha) = k. Returns a list or tuple of n classical one-qbit
    states (either |0> or |1>), such that the corresponding n-bit string delta
    usually satisfies f(delta) = 1.'''
```

B. In qAlgorithms.py, add the following function, and use it to test your `grover`.

```
def groverTest(n, k):
    # Pick k distinct deltas uniformly randomly.
    deltas = []
    while len(deltas) < k:
        delta = qb.string(n, random.randrange(0, 2**n))
        if not delta in deltas:
            deltas.append(delta)
    # Prepare the F gate.
    def f(alpha):
        for delta in deltas:
            if alpha == delta:
                return (1,)
        return (0,)
    fGate = qg.function(n, 1, f)
    # Run Grover's algorithm up to 10 times.
    qbits = grover(n, k, fGate)
    bits = tuple(map(qu.bitValue, qbits))
    j = 1
    while (not bits in deltas) and (j < 10):
        qbits = grover(n, k, fGate)
        bits = tuple(map(qu.bitValue, qbits))
        j += 1
    if bits in deltas:
```

```
        print("passed groverTest in " + str(j) + " tries")
    else:
        print("failed groverTest")
        print("    exceeded 10 tries")
        print("    prediction = " + str(bits))
        print("    deltas = " + str(deltas))
```