This assignment is project work due at the end of the term.

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

```
def simon(n, f):
    '''The inputs are an integer n >= 2 and an (n + (n - 1))-qbit gate F
    representing a function f: {0, 1}^n -> {0, 1}^(n - 1) hiding an n-bit
    string delta as in the Simon (1994) problem. Returns a list or tuple of n
    classical one-qbit states (each |0> or |1>) corresponding to a uniformly
    random bit string gamma that is perpendicular to delta.'''
```

B. In qAlgorithms.py, paste the following test function. Replace the `pass` line with code that uses Simon's algorithm to make a `prediction` for δ. The function `reduction` in qBitStrings.py should help.

```
def simonTest(n):
    # Pick a non-zero delta uniformly randomly.
    delta = qb.string(n, random.randrange(1, 2**n))
    # Build a certain matrix M.
    k = 0
    while delta[k] == 0:
        k += 1
    m = numpy.identity(n, dtype=int)
    m[:, k] = delta
    mInv = m
    # This f is a linear map with kernel {0, delta}. So it's a valid example.
    def f(s):
        full = numpy.dot(mInv, s) % 2
        full = tuple([full[i] for i in range(len(full))])
        return full[:k] + full[k + 1:]
    gate = qg.function(n, n - 1, f)
    pass
    if delta == prediction:
        print("passed simonTest")
    else:
        print("failed simonTest")
        print("    delta = " + str(delta))
        print("    prediction = " + str(prediction))
```