This assignment is entirely Python programming. Start from a copy of `qc.py` from our Moodle site. Remember that this file implements one-qbit states as two-dimensional `numpy` arrays and one-qbit gates as 2×2 `numpy` matrices. You may optionally add to this file the one-qbit `measurement` function that you wrote in an earlier assignment. Now we are going to start adding a bunch of two-qbit functionality. A two-qbit state will be a four-dimensional `numpy` array, and a two-qbit gate will be a 4×4 `numpy` matrix.

A. Add constant 4×4 matrices called `cnot` and `swap` to implement those two-qbit gates. (I suggest that you add them near the other constants.)

B. Write a function `first` precisely according to the following specification. As input it takes a two-qbit state $|\psi\rangle$. It performs a partial measurement of the first qbit. As output it returns a pair (a Python tuple or list of two items) consisting of a classical one-qbit state (either $|0\rangle$ or $|1\rangle$) and a one-qbit state.

C. Add the following function to your `qc.py`. On your own, use it as part of your testing that your implement of `first` works. But you will probably want to do more testing than just this. Consider writing other "`firstTest...`" functions.

```
def firstTest():
    # Constructs an unentangled two-qbit state |0> |psi> or |1> |psi>,
    # measures the first qbit, and then reconstructs the state.
    print("One should see 0s.")
    psi = uniform(1)
    state = tensor(ket0, psi)
    meas = first(state)
    print(state - tensor(meas[0], meas[1]))
    psi = uniform(1)
    state = tensor(ket1, psi)
    meas = first(state)
    print(state - tensor(meas[0], meas[1]))
```

D. Similarly, write a function `last` which takes as input a two-qbit state, performs a partial measurement on the second qbit, and returns a pair consisting of a one-qbit state and a classical one-qbit state. Also write one or more "`lastTest...`" functions.

E. Write a function `application` which takes as input a two-qbit gate U and a two-qbit state $|\psi\rangle$, and returns as output the two-qbit state $U|\psi\rangle$. (Hint: This task can be accomplished with a single `numpy` function call.) Test your implementation, but I don't need to see your tests.

F. Write a function `tensor` which computes the tensor product of one-qbit states and gates. I mean, when it is given states $|\psi\rangle$ and $|\phi\rangle$, it returns the state $|\psi\rangle \otimes |\phi\rangle$, and when it is given gates U and V, it returns the gate $U \otimes V$. Test your implementation, but I don't need to see your tests.

To clarify: You will hand in your copy of `qc.py` containing the two constants (`cnot` and `swap`) and six functions (`first`, `firstTest`, `last`, `lastTest`, `application`, `tensor`) specified in this assignment. You do not need to hand in any other test functions, although you may. You do not need to hand in any test results.

Your `qc.py` will be graded by importing it into another program that runs tests using your functions. So make sure that your functions are well-tested and that they exactly conform to the specifications.