CMPT 476/776: Introduction to Quantum Algorithms
Assignment 2

Due **February 5th, 2026 at 11:59pm on crowdmark**
Complete individually and submit in PDF format.

Question 1 [5 points]: Entanglement

Consider the following two-qubit gate:

$$\Delta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A two-qubit gate is **entangling** if it maps **at least one** separable state $|\phi\rangle \otimes |\psi\rangle$ to an entangled state, and is **separable** if it can be written as a tensor product of 2x2 matrices.

1. Is the Δ gate entangling? Give a proof for your answer.
2. Is the Δ gate separable? Give a proof for your answer.
3. How might you interpret the effect of the Δ gate on a two-qubit state?

Question 2 [4 points]: Pauli operators

Recall the definition of the I, X, Z, and Y gates:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

These are known as the *Pauli matrices* or gates.

1. Compute the matrices $X \otimes Z$ and $Z \otimes X$
2. Show that the non-identity Pauli matrices anti-commute: that is, $UV = -VU$ for every pair of X, Y, and Z matrices where $U \neq V$
3. Show that the Pauli matrices I, X, Z, Y are linearly independent
4. Show that the Pauli matrices form a basis for the space of 2×2 complex-valued matrices.

Question 3 [9 points]: Non-local games

In this question, we're going to study another non-local game involving 3-parties, or 3 qubits. First let $|\psi\rangle = \frac{1}{2}(|000\rangle - |110\rangle - |011\rangle - |101\rangle)$.

1. Give a 3-qubit circuit U consisting of X, H, and $CNOT$ gates such that
$$U\left(\frac{1}{\sqrt{2}}|000\rangle - \frac{1}{\sqrt{2}}|111\rangle\right) = |\psi\rangle.$$
2. Show that a partial measurement of any qubit of the $|\psi\rangle$ state leaves an entangled state in the remaining 2 qubits.
3. Compute the parity $a \oplus b \oplus c = a + b + c \mod 2$ of the measurement results if
   (a) All qubits are measured in the $\{0, 1\}$ basis.
   (b) Qubits 0 and 1 are measured in the $\{|+\rangle, |-\rangle\}$ basis and qubit 2 in the $\{0, 1\}$ basis.
   (c) Qubits 0 and 2 are measured in the $\{|+\rangle, |-\rangle\}$ basis and qubit 1 in the $\{0, 1\}$ basis.
   (d) Qubits 1 and 2 are measured in the $\{|+\rangle, |-\rangle\}$ basis and qubit 0 in the $\{0, 1\}$ basis.

   In the $\{|+\rangle, |-\rangle\}$ basis, interpret the measurement result "+" as 0 and "-" as 1.
4. Denote the measurement result of qubit i in the $\{0, 1\}$ basis by a_i, and in the $\{|+\rangle, |-\rangle\}$ basis by b_i. Is it possible that each measurement result a_i and b_i has a **pre-determined** value which is **independent** of which basis the other qubits are measured in?

   Hint: Add up the 4 cases in the previous question mod 2. Is anything wrong?
5. Give a **perfect** quantum strategy (i.e. a strategy involving a shared pre-entangled state which **wins 100% of the time**) for the following 3 player game.
   - Alice, Bob, and Charlie are each given one bit x, y, and z respectively with the constraint that $x \oplus y \oplus z = 0$.
   - Alice, Bob, and Charlie each return a single bit a, b, c respectively, and they win if $a \oplus b \oplus c = x \vee y \vee z$.

   To get you started, use the state $|\psi\rangle$ from the first part of this question as the initial shared state.

Question 4 [4 points]: Partial measurement and mixed states

Let
$$|\psi\rangle = \frac{i\sqrt{2}}{\sqrt{3}}|00\rangle + \frac{1}{\sqrt{3}\sqrt{2}}|01\rangle + \frac{\sqrt{2}}{2\sqrt{3}}|10\rangle.$$

1. Calculate the probabilities of measuring 0 or 1 in the first qubit, and the resulting normalized state vector in each case.
2. Write the mixed state obtained after measuring the first qubit as a density matrix ρ.
3. Compute the partial trace $\mathrm{Tr}_A(\rho)$ obtained by tracing out the first qubit of ρ.
4. Compute the partial trace $\mathrm{Tr}_B(\rho)$ obtained by tracing out the second qubit of ρ.

Question 5 [8 points]:

In this question you're going to write a tiny simulator for quantum circuits, which explicitly performs — on a classical computer such as yours — the linear algebraic calculations that quantum circuits represent. We'll use Python with Numpy to provide some data structures and randomization routines. Starter code and some simple sanity checks are provided in the file `simulator.py`, along with the constant and function stubs you are tasked with filling in.

Tasks:

1. Fill out the definitions of the `ket0` and `ket1` states.
2. Fill out the defintions of the `X`, `Y`, `Z`, `H`, and `CNOT` gates.
3. Fill out the definitions of the `normalize` and `tensor` operators. **You may not use numpy's `normalize` or `kron` functions**. `tensor` should allow tensor products of vectors and matrices of arbitrary dimensions.
4. Fill out the definition of `measure`, which takes a state vector and measures the indicated qubit number (starting from 0), and returns a pair $(result, |\psi\rangle)$ consisting of the numerical result of the measurement, $result$, and the resulting (normalized) state vector $|\psi\rangle$.
5. Test it out on some larger circuits, e.g. by applying some gates to a large state like $|+\rangle^{\otimes 20}$, the tensor product of 20 copies of $|+\rangle$. What do you notice? Do you think classical computers can efficiently simulate arbitrary quantum circuits? How can you make this simulation more scalable?

Notes:

- Represent the computational basis in big-endian (most significant bit first). A helper function, `toBits` is provided which decomposes an integer i into a length n list of bits in big-endian.
- Vectors are most conveniently represented as matrices (i.e. 2d arrays) where there is only a single column. You may use a 1d array instead, but `tensor` becomes slightly more involved.
- `numpy.random` has a function, `choice`, which allows specification of a probability distribution for the random choice and will hence be helpful.
- The conjugate of a complex number `c` can be obtained by `c.conjugate()`
- Part of your mark will be auto-graded, so test your code thoroughly.