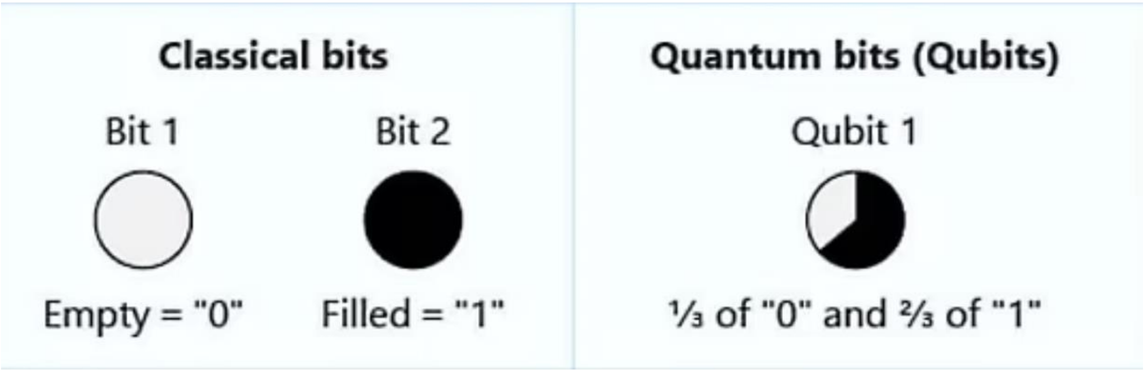
Object-Oriented Programming
Programming Project #1

Classical Bit vs Quantum Bit

- Classical bit:
  - Basic unit of information in traditional computing
  - Either 0 or 1
- Quantum bit (i.e., qubit):
  - Basic unit of information in quantum computing
  - Superposition of 0 and 1; a certain probability of being a 0 and a certain probability of being a 1



Classical Bit vs Quantum Bit

- Classical bit:
  - Basic unit of information in traditional computing
  - Either 0 or 1
- Quantum bit (i.e., qubit):
  - Basic unit of information in quantum computing
  - Superposition of 0 and 1; a certain probability of being a 0 and a certain probability of being a 1
- Superior computing power:
  - Finding the prime factors of a 2048-bit number
  - Take million of years on a traditional computer
  - Need only minutes on a quantum computer

Their Physical Implementations

- Classical bit:
  - Silicon-based chips
- Quantum bit (i.e., qubit):
  - Trapped ions, photons, artificial or real atoms, or quasiparticles
  - Some needs their qubits to be kept at temperatures close to absolute zero

Quantum Notation (Dirac Notation)

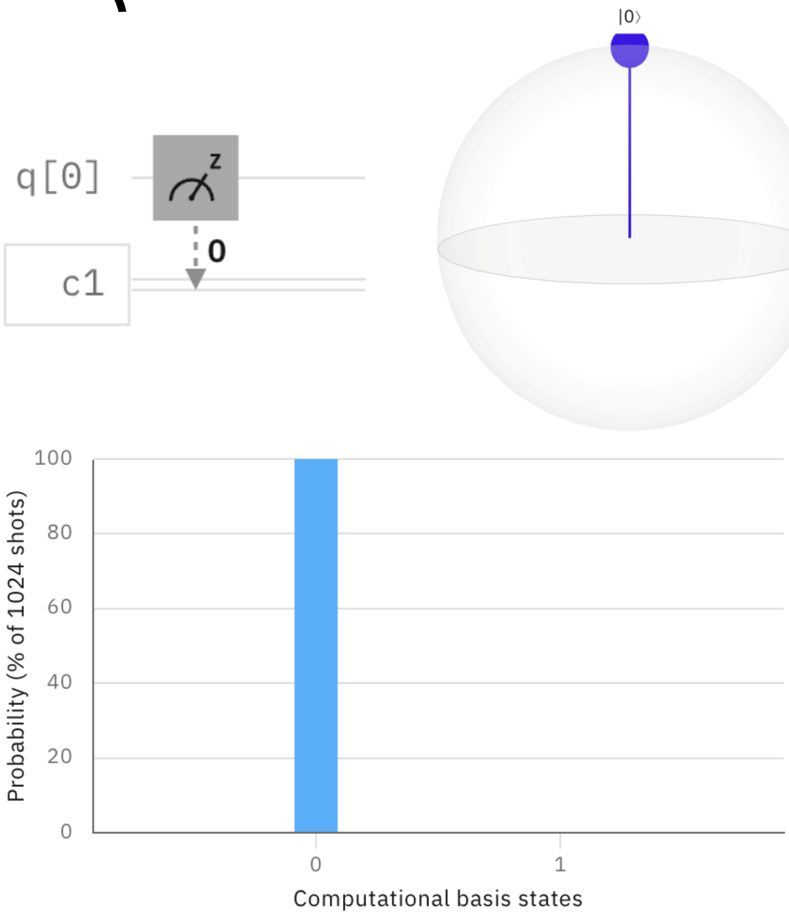
- ket: $|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix}$
  - Represent a state of some quantum system, where n is the dimension and $\psi_1, \ldots, \psi_n$ are complex numbers
- For a qubit (i.e., 2-dimension):
  - Orthogonal basis: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
  - Any state vector $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ can be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$

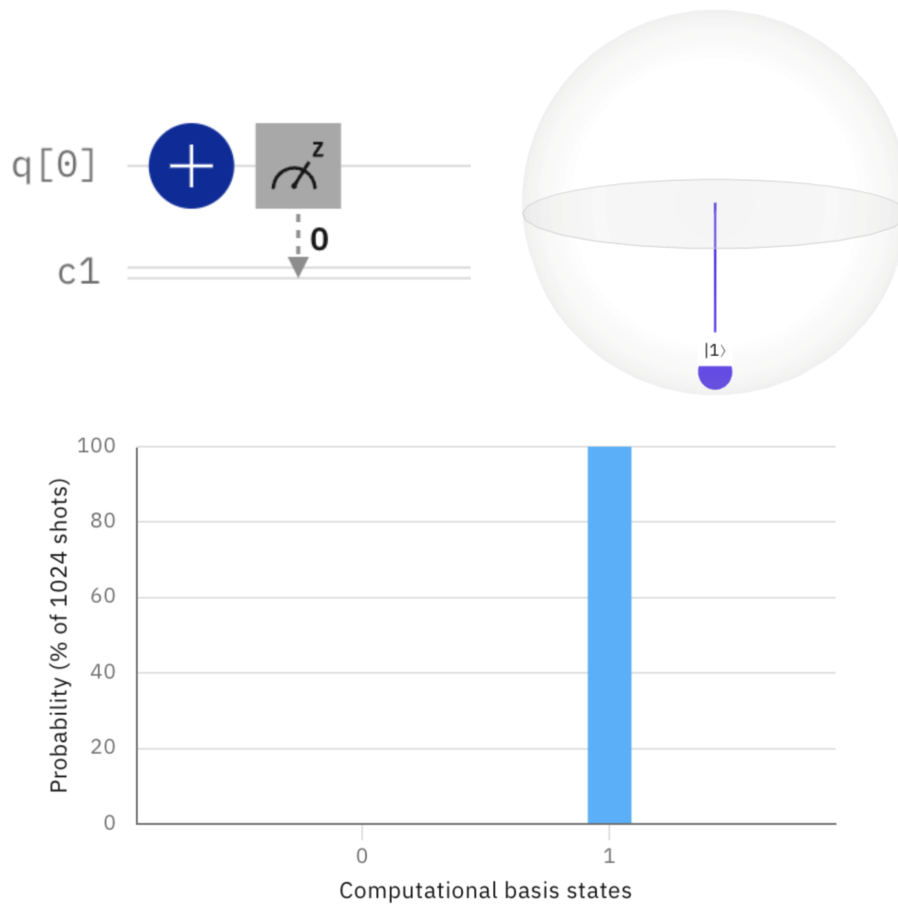
Some Basic Quantum Gates

Operator	Gate(s)		Matrix
Pauli-X (X)	$\boxed{\mathbf{X}}$	$\oplus$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	$\boxed{\mathbf{Y}}$		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	$\boxed{\mathbf{Z}}$		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	$\boxed{\mathbf{H}}$		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

- For a qubit:
  - $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle$
  - $X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}$
  - $X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$

Quantum Gate and Quantum Circuit



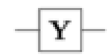
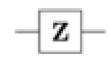
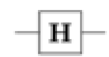
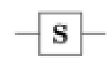
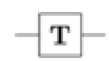
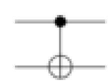
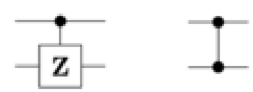


$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$
$$= |1\rangle$$

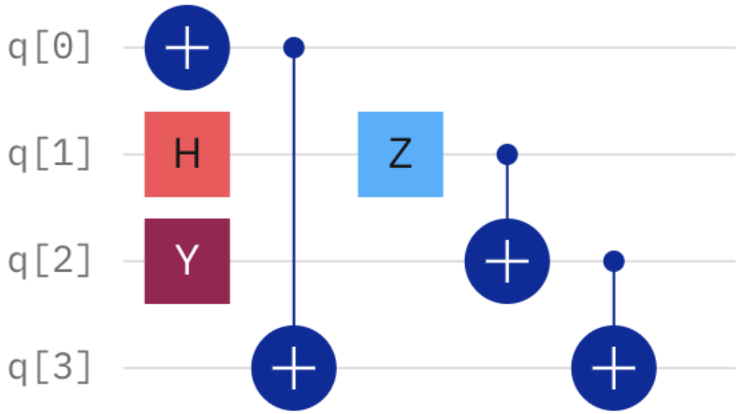
IBM Quantum Lab https://quantum.ibm.com/lab

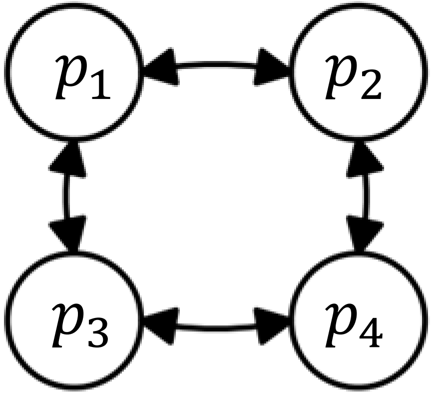
More Quantum Gates

	Operator	Gate(s)	Matrix
One-qubit gate	**Pauli-X (X)**		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
	Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
	Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
	Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
	Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
	$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
two-qubit gate	**Controlled Not (CNOT, CX)**		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
	Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
	SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Quantum Circuit and Quantum Processor

- Quantum circuit:
  A diagram representing a quantum program
- Physical quantum device:
  A topology showing the coupled physical qubits

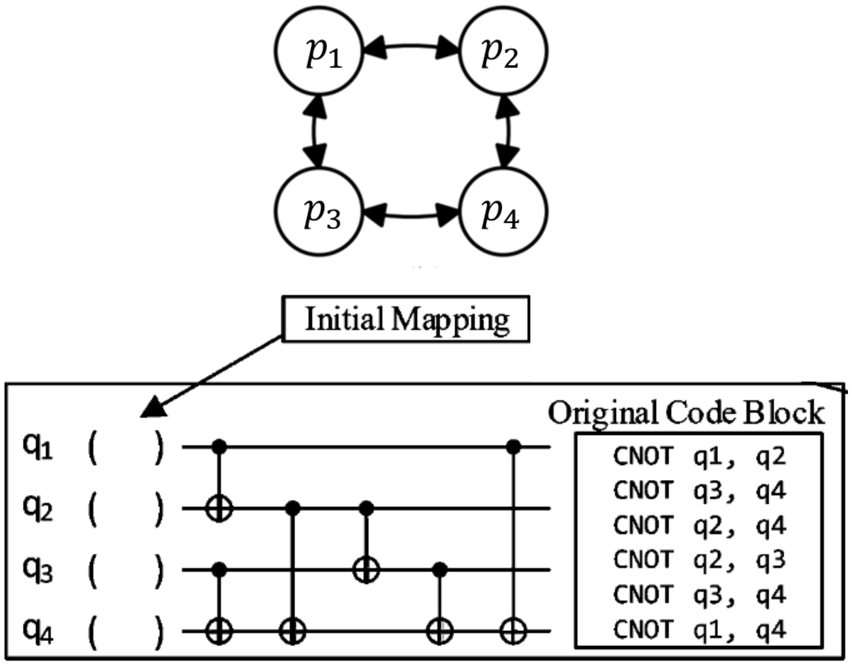




IBM Quantum Lab https://quantum.ibm.com/lab

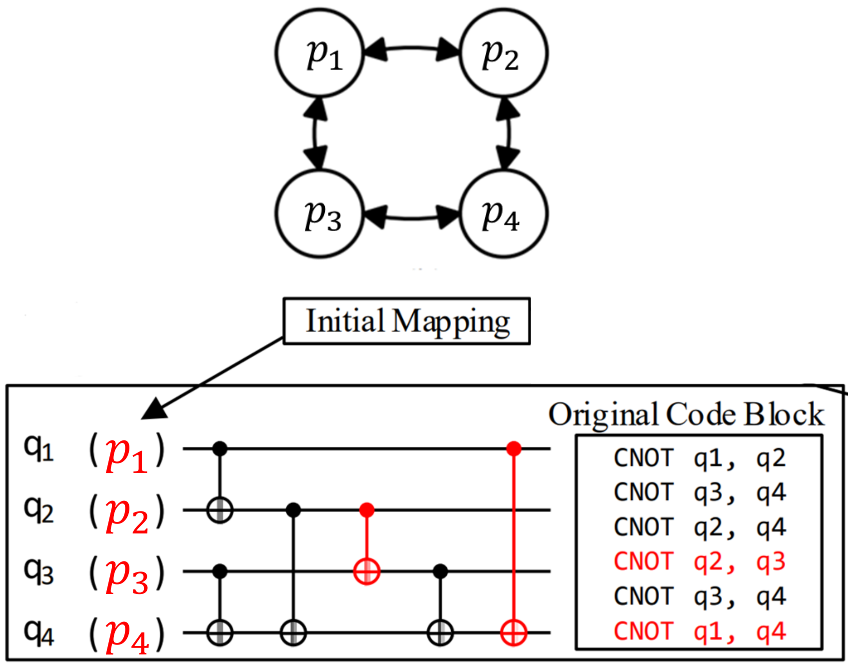
Quantum Circuit and Quantum Processor

- Logical qubits and original circuit:
- Physical qubits and hardware-compliant circuit



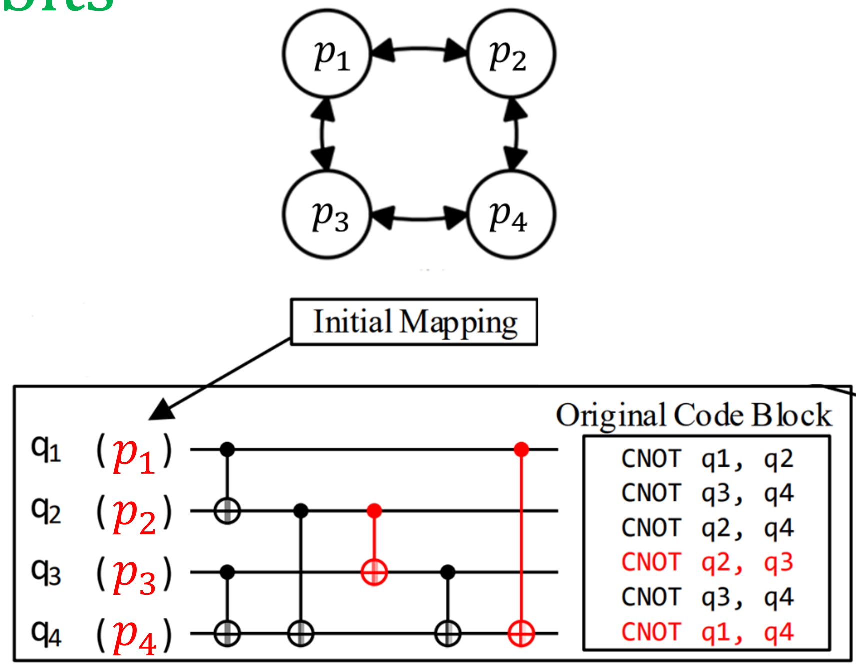
Quantum Circuit and Quantum Processor

- Logical qubits and original circuit:
- Physical qubits and hardware-compliant circuit
- Find an initial logical-to-physical qubit mapping



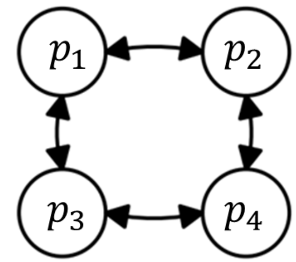
Quantum Circuit and Quantum Processor

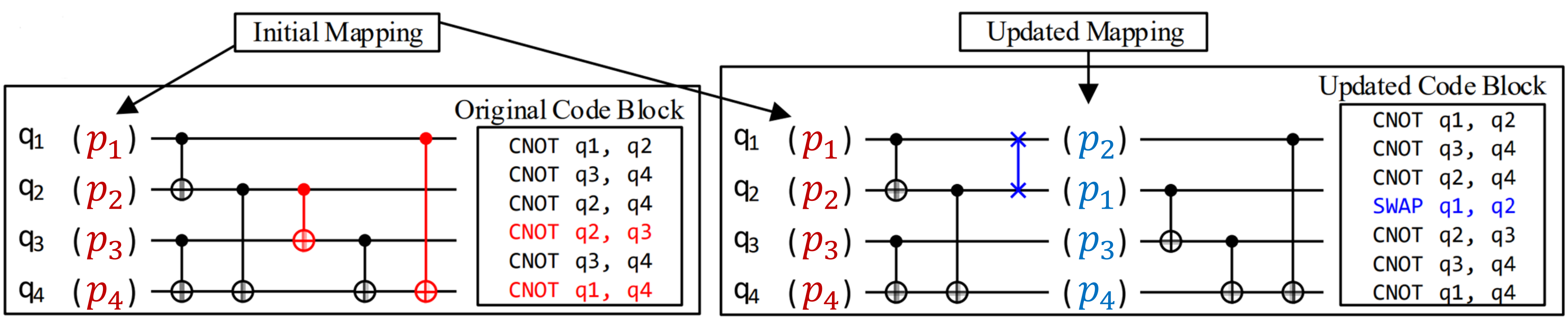
- However, it might not be always feasible
- Goal: Ensure every two logical qubits in a two-qubit gate are always mapped to two coupled physical qubits



Quantum Circuit and Quantum Processor

- Two main procedures:
- Initial logical-to-physical qubit mapping
- Intermediate mapping transition
  - Swap the mapped physical qubits





Requirements

- You can make all member variables public
- Try to define your own classes
  - class PQB (physical qubit), including ID, LQBID, nPQBIDs
  - class LQB (logical qubit), including ID, PQBID
  - class gate, including precedence, LQBIDs
  - …
- Try to use pair, vector, or map

Programming Project #1: Qubit Mapping and Routing Problem

- Input:
  - # logical qubits, # physical qubits, # physical links
  - 2-qubit gates and their precedence in the logical circuit
  - The topology of the physical quantum device
- Procedure:
  - Compute the initial mapping
  - Compute the swap sequence
- Output:
  - The initial mapping for each logical qubit
  - The gate sequence including additional swaps

The Competition

- The grade is inversely proportional to # swaps
- Basic: 60 (deadline)
  - Feasible solution
- Being a coding assistant (superb deadline)
  - +10
- Performance ranking (decided after the deadline)
  - [0%, 30%) (bottom): +0
  - [30%, 50%): + 5
  - [50%, 75%): + 10
  - [75%, 85%): + 15
  - [85%, 90%): + 20
  - [90%, 95%): + 25
  - [95%, 100%] (top): + 30

The Competition Rules

- Note that you cannot use brute-force algorithm
- Your solution should be deterministic on our server
  - E.g., the random seed & the number of iterations are fixed

Input Sample:

use cin

Format:

#logQubits #gates #precedences #phyQubits #phyLinks

...

gateID logQubitID1 logQubitID2

...

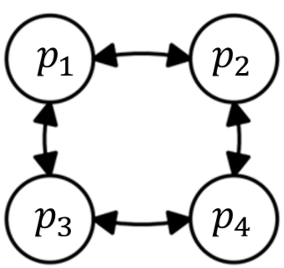
precedenceID gateID1 gateID2

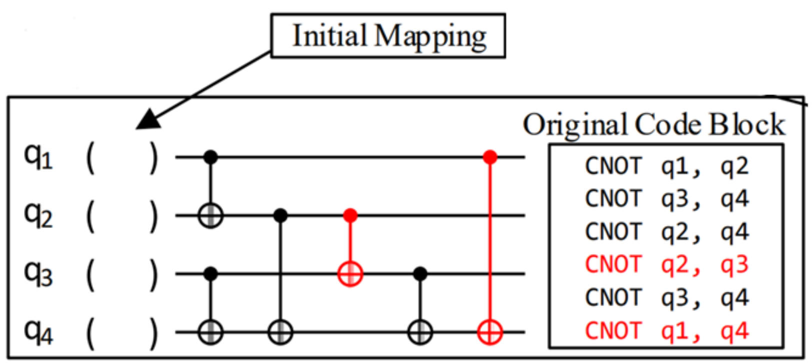
...

phyLinkID phyQubitID1 phyQubitID2

...

Note that all ID start from 1






```
4 6 5 4 4
1 1 2
2 3 4
3 2 4
4 2 3
5 3 4
6 1 4
1 1 3
2 2 3
3 3 4
4 4 5
5 5 6
1 1 2
2 1 3
3 2 4
4 3 4
```

Output Sample (not optimal):
use cout

Format:

...

logQubitID mappedphyQubitID

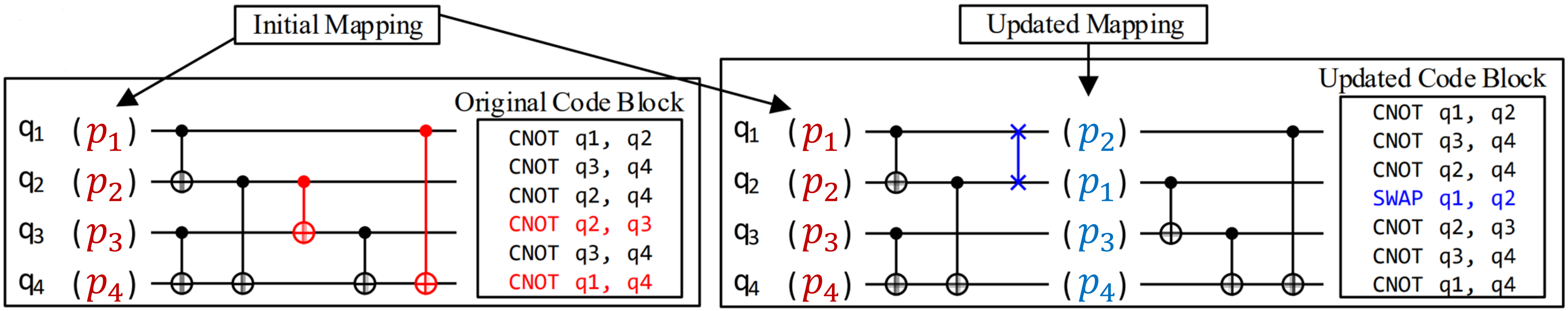
...

Updated Code Block

e.g.,

```
1 1
2 2
3 3
4 4
CNOT q1 q2
CNOT q3 q4
CNOT q2 q4
SWAP q1 q2
CNOT q2 q3
CNOT q3 q4
CNOT q1 q4
```

Note that all ID start from 1



Note

- Superb deadline: 4/4 Thu
- Deadline: 4/11 Thu
- Pass the test of our online judge platform
- Submit your code to E-course2
  - The file name should be ``OOP_HW1_studentID.cpp''
- Demonstrate your code remotely with TA
- **C++ Source code (only C++; compiled with g++)**
  - **Include C++ library only (i.e., no stdio, no stdlib, …)**
  - **Please use new and delete instead of malloc and free**
- Show a good programming style