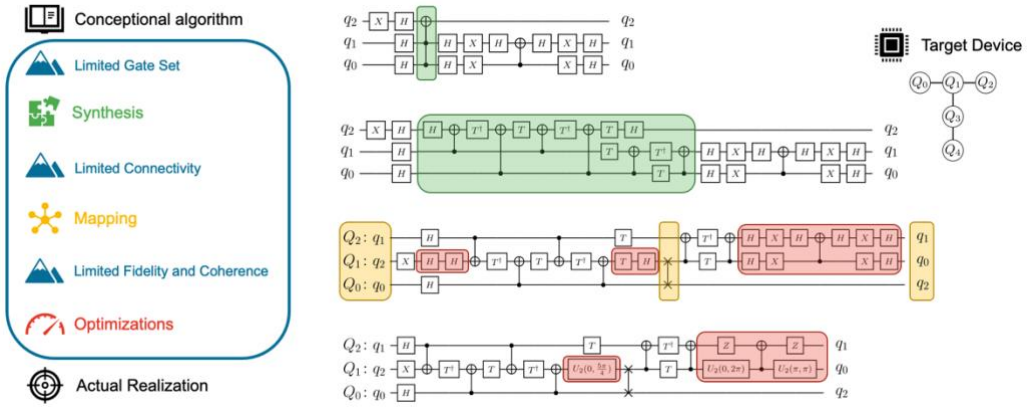
Emerging Technologies 2021 – Programming Task

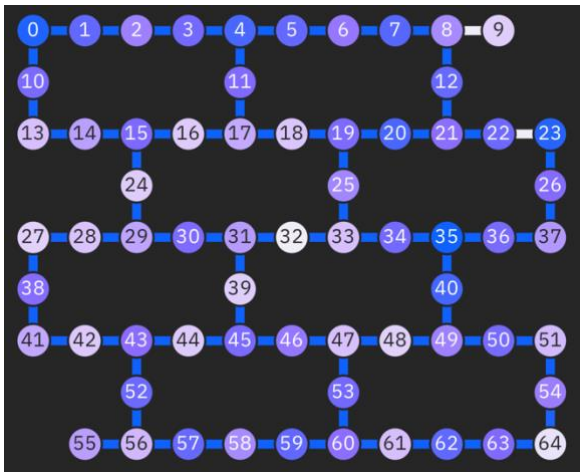
In this year's programming task, we challenge you to develop an efficient mapping solution for quantum circuits. Quantum circuit mapping is a subtask in the overall compilation flow of quantum circuits which is required to execute a conceptual quantum algorithm on an actual device. The following figure gives a rough overview of which obstacles need to be overcome when compiling a quantum circuit.



To this end, the purpose of the mapping task is to make sure that any two-qubit gate of the algorithm is applied to qubits that are connected on the targeted device (where the connectivity is described by a so-called coupling map as shown on the right-hand side in the figure above). This is commonly achieved by establishing a mapping between the algorithm's logical qubits and the device's physical qubits. Since it is quite rare that such a static assignment exists that makes every gate executable, SWAP operations are introduced in the circuit, which allow to dynamically change the mapping. Due to quantum computers being inherently affected by noise and decoherence, the goal is to realize this mapping with the least possible overhead.

You will be given instances of various quantum algorithms that have been realized using the default IBM gate-set consisting of single-qubit X, $R_Z(\lambda)$, $\sqrt{X}$, and two-qubit $CNOT$ gates. For an explanation of these gates, see https://quantum-computing.ibm.com/composer/docs/iqx/operations_glossary.

Your task is to develop a mapper, that takes these circuits and transforms them to a representation so that they can be executed on the 65-qubit IBMQ Brooklyn device, that is specified by the following coupling map:



In particular, you will be given *.qasm* files describing the unmapped circuits and are asked to produce *.qasm* files containing the circuits that have been mapped by your solution to the IBM architecture described above. Your solution will be scored based on the cumulative number of gates needed to realize all the circuits with the following costs per gate:

- Since $R_Z(\lambda)$ gates can be implemented virtually in hardware via frame changes (i.e., at zero error and duration) they are assigned a cost of 0.
- Any $\sqrt{X}$ or X gate is assigned a cost of 1.
- Since the error rate of $CNOT$ gates typically is around $10 \times$ the error rate of single-qubit operations, they are assigned a cost of 10.
- Since a $SWAP$ gate is realized as a sequence of 3 $CNOTs$, $SWAPs$ are assigned a cost of 30.

The solution with the minimal number of overall gates wins the competition.

A few restrictions and conditions apply:

- The resulting circuits need to be valid *.qasm* files, i.e., it must be possible to parse the files, e.g., using IBM's Qiskit.
- The resulting circuits must conform to the prescribed coupling map, i.e., your solution must yield valid mappings.
- The resulting circuits must not contain any other gates than those provided by the IBM gate-set with the exception of the $SWAP$ gate, which is also permitted.
- The resulting circuits must realize the same functionality as the originally provided circuits, i.e., the way the circuits work must not be altered.
- Your proposed solution should be generic, i.e., you should not incorporate any knowledge about the provided circuits into your mapping technique.
- You must be able to explain your solution and are not allowed to plagiarize.

These rules are subject to change during the competition if any uncertainties or problems arise.

We do expect you to provide your own solutions to the mapping problem. However, for starters, the following Python snippet can be used to read in a *.qasm* file, map the circuit, and dump the resulting *.qasm* representation using IBM's Qiskit:

```
from qiskit import QuantumCircuit, transpile
from qiskit.test.mock.backends import FakeBrooklyn

path = './emtec21/benchmarks/original/circuit.qasm'
qc = QuantumCircuit.from_qasm_file(path=path)
qc_transpiled = transpile(qc, backend=FakeBrooklyn())
filename = './emtec21/benchmarks/mapped/circuit.qasm'
qc_transpiled.qasm(filename=filename)
```

If you want, you can develop your whole solution in IBM Qiskit, for which you might want to have a look at
https://qiskit.org/documentation/tutorials/circuits_advanced/04_transpiler_passes_and_passmanager.html#Implementing-a-BasicMapper-Pass.
However, in principle, you are free to use any language of your choice.