1. Task one – Shor's Algorithm

(a) Task Target –

a. Basic
   i. Understand and practice Shor's Algorithm.
   ii. Learn the optimization methods.

b. Advanced –
   i. Try to debug the program; Conclude the methods and illustrate the difficulties.
   ii. Try to accelerate repeated circuits execution.
   iii. Try to introduce parallel computation and report difficulties.
   iv. Try to predicate partial execution results.
   v. Try to cross-platform calls, e.g. call your Qiskit program from Matlab.
   vi. Try to adapt your coding to large numbers.
   vii. Try to learn the decomposition algorithms.

(b) Tools requirements – Circuits part (operations and classical programming) of Q# or IBM Qiskit with Python features.

(c) Do NOT-

a. Please do not use any inherited "high-level Quantum functions" especially oracle related and phase estimation related. They are the key practice of this task.
   The following functions needs to be strictly avoided:

Microsoft.Quantum.Canon.RobustPhaseEstimation
Microsoft.Quantum.Canon.QuantumPhaseEstimation
Quantum.Canon.ExpMod
Microsoft.Quantum.Canon.DiscreteOracle
QISKit.Terra

   The following functions may need to be avoided, if you can:

Microsoft.Quantum.Canon.ModularMultiplyByConstantLE

b. Please do not use quantum features "do…repeat" structure (Q#) or "q_if" structure (Qiskit) unless you confirm you have clearly knowledge how to implement this part in quantum circuits.

(d) Please DO-

a. We recommend module programming, i.e., separate classical processing, "Order-finding" and "Quantum Fourier Transformation" etc. into modules.
b. We recommend use proper comments in your programs.
c. We recommend use "Unit Test" for your modules and keep these tests in your assignments.

(e) Input and Output (Even this practicedoes not have strict input and output requirements, we suggest the following criteria)-

a. The input is
   i. Only a random integer number ranged from 1 to 31.
   ii. Or with a probability ε, s.t., the programs execute *Poly (log_2N)* steps and produce the factors with a probability greater or equal to (1- ε). (Advanced Tasks)
   iii. Or with the Clifford+T group, where

$$\mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \mathrm{S} = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \mathrm{CNOT} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}, \mathrm{T} = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}.$$

   (Advanced Tasks)

b. The output contains several parts, including
   i. The quantum circuits described by your preferred format such as XML, json…Please note you can design your file format for supporting quantum features. If you try the advanced tasks, you should consider the circuits and related decomposition algorithm which are described by Clifford+T group with a fixed depth.
   ii. And/or quantum circuits by your prefer language description, e.g., generate Q#, IBM Qiskit statements only including quantum operations without further functions calls.
   iii. A tester (Manifest) which can execute your designed algorithm and output the simulation results and the circuit descriptions.
   iv. A description file shows your designing details and your create.

(f) You can use the following link as references.

a. https://github.com/Microsoft/Quantum/blob/master/Samples/IntegerFactorization/Shor.qs
b. https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=004-Quantum_Algorithms~2F110-Shor%27s_algorithm
c. https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/004-Quantum_Algorithms/100-Quantum_Phase_Estimation.html