CS166 WI24: Homework 5 (Due Friday March 1 11:59pm)

※ Problem 1: n-qubit Hadamard

In lecture, we encountered the following identity for the n-qubit Hadamard.

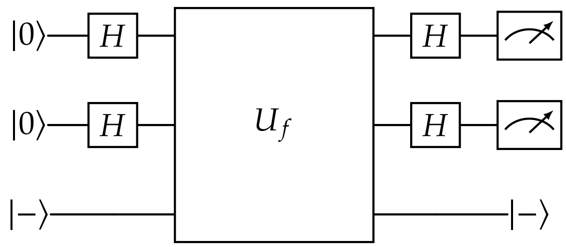
Proposition 1.1 (n-qubit Hadamard)**.** Let $x = x_1 x_2 \cdots x_n$ be the binary expansion of x. In other words, x_i is the i-th bit of x when x is written in binary. Then, we have the following identity:

$$
\begin{aligned}
H^{\otimes n} |x\rangle &= H|x_1\rangle \otimes H|x_2\rangle \otimes \cdots \otimes H|x_n\rangle && (1)\\
&= \frac{(|0\rangle + (-1)^{x_1}|1\rangle)}{\sqrt{2}} \otimes \cdots \otimes \frac{(|0\rangle + (-1)^{x_n}|1\rangle)}{\sqrt{2}} && (2)\\
&= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle && (3)
\end{aligned}
$$

where $x \cdot y$ is the bit wise dot product of x and y (i.e., $x \cdot y = x_1 y_1 + \cdots + x_n y_n$).

1. Consider the case where $n = 3$, and $x = 101$.
   (a) Write down equations (1), and (2) for this case explicitly.
   (b) Distribute the tensor product you have from equation (2) and verify that each coefficient matches equation (3).
2. Consider the case where $n = 4$, and $x = 0000$.
   (a) What is equation (2) in this instance?
   (b) What is the probability that we measure 0010 if we measure the four qubits in the standard basis?
   (c) More generally, what can we say about the distribution of outputs when we measure this state in the standard basis?
3. Prove the proposition.

※ Problem 2: Deutsch-Josza Explicit Example



Let f be a function that takes 2 bits as inputs and outputs a single bit. The function takes the two bits it received, adds them all together, and outputs the answer mod 2.

1. Is this function constant or balanced?
2. (Particular instance) What is $U_f\,|11\rangle\,|-\rangle$?
3. Now we begin analyzing the full algorithm. Starting from $|00\rangle\,|-\rangle$ as we have in the diagram, what is the state of the algorithm after the H gates? Use the n-qubit Hadamard identity.
4. (Continue from 3) What is the state of the algorithm after the U_f gate is applied?
5. (Continue from 4) What is the state of the algorithm after the second layer of H gates? Don't use summation notation, and explicitly write out the coefficients.
6. (Continue from 5) What are the possible measurement results and their corresponding probabilities?

※ Problem 3: Simon's problem example

The function $f : \{0,1\}^3 \rightarrow \{0,1\}^3$ is defined as follows:

$$\begin{aligned} f(000) &= 110 \quad f(100) = 001 && (4)\\ f(001) &= 001 \quad f(101) = 110 && (5)\\ f(010) &= 000 \quad f(110) = 010 && (6)\\ f(011) &= 010 \quad f(111) = 000 && (7)\\ & && (8) \end{aligned}$$

1. The inputs are paired so that for $x \neq y$, $f(x) = f(y)$ if and only if $x = y \oplus s$ for some fixed s. Can you figure out s by inspection?

2. In this question, you will simulate Simon's algorithm for the function f. The unitary operator U_f maps $|x\rangle\,|000\rangle$ to $|x\rangle\,|000 \oplus f(x)\rangle$, where x is a 3-bit string. The operator $\oplus$ is bit-wise addition, mod 2. Write down the state of the algorithm after each step. Use the n-qubit Hadamard identity.
   - Start with $|000\rangle\,|000\rangle$.
   - Apply $H^{\otimes 3} \otimes I^{\otimes 3}$.
   - Apply U_f.

3. Based on your answer from the previous problem, what are the possible states we can see if we measure the last three qubits? What are the corresponding probabilities?

4. Suppose you measured 110 in the previous step. What is the state of the algorithm?

5. (Continue from 4) Apply $H^{\otimes 3}$ on the first three qubits. What are the possible measurement outcomes? Use the n-qubit Hadamard identity.

6. (Continue from 5) Verify that every string x that is a possible outcome of the last measurement satisfies $x \cdot s = 0 \bmod 2$.

※ Problem 4: Simon's problem implementation

```
import numpy as np

f = {
    '000': '110',
    '001': '001',
    '010': '000',
    '011': '010',
    '100': '001',
    '101': '110',
    '110': '010',
    '111': '000'
}

N = 2 ** 6

U_f = np.identity(N, dtype=complex)

for input_state, output_state in f.items():
    input_index = int(input_state[::-1], 2) + int('000', 2) * 2**3
    output_index = int(
        input_state[::-1], 2) + int(output_state[::-1], 2
        ) * 2**3
    U_f[[input_index, output_index]] = U_f[[output_index, input_index]]
```

Copy and paste the above code snippet into your environment. You can apply a U_f gate by the code

```
qc.unitary(U_f, range(6), label='Uf')
```

Implement the algorithm you analyzed in Problem 3. Can you recover the secret string using your code?