# CS419/CS939: Quantum Computing – Assignment 3

## Problem 1

**(35 points)** Let $F\colon \{0,1\}^n \to \{0,1\}$ be the logical OR function (that is, $F(x_1, x_2) = 0$ if and only if $x_1 = x_2 = 0$).

1. Covert $F$ into a the $\{-1,1\}$-valued function $f\colon \{0,1\}^2 \to \{-1,1\}$.
2. Write the quantum state $|f\rangle$.
3. Compute the Fourier spectrum of $f$; that is $\hat{f}(x)$ for each $x \in \{0,1\}^2$.
4. Express $f$ in its Fourier expansion.
5. Write the quantum state $\left|\hat{f}\right\rangle$.
6. What will we get if we apply $H^{\otimes 2}$ to $|f\rangle$?
7. What will we get if we apply $H^{\otimes 2}$ to $\left|\hat{f}\right\rangle$?

## Problem 2

**(35 points)** Let $f_0\colon \{0,1\}^n \to \{-1,1\}$ be a function such that $\hat{f}(00\cdots 0) = \sqrt{2/3}$, and let $f_1\colon \{0,1\}^n \to \{-1,1\}$ be a function such that $\hat{f}(11\cdots 1) = \sqrt{2/3}$. Design and analyse an $O(1)$-query quantum algorithm that can distinguish between $f_0$ and $f_1$ with probability at least 99%.

## Problem 3

**(30 points)** Let $f : [\sqrt{N}] \times [\sqrt{N}] \to 0, 1$ be a function representing a $\sqrt{N} \times \sqrt{N}$ Boolean matrix. Design an $O(N^{3/4}\log(N))$-query quantum algorithm that checks whether there exists at least one row that consists of only ones; that is, there exists $x$ such that $f(x, y) = 1$ for all $y$. What is the *time* complexity of your algorithm?

## Problem 4 – BONUS

**(10 additional points, max mark is still 100 though...)**

1. Watch the following Qiskit tutorials:
   `https://www.youtube.com/playlist?list=PLOFEBzvs-Vvp2xg9-POLJhQwtVktlYGbY`
2. Implement a quantum algorithm of your choice (e.g., Deutsch, Deutsch-Jozsa, Bernstein-Vazirani, Fourier Sampling, Grover's search, Simon's algorithm, etc.) using Qiskit.

## Some hints

1. Say algorithm $A$ makes $q$ queries and errs with probability at most $1/3$. Then for every $m$, there exists an algorithm $A'$ that makes $O(q \log(m))$ queries and errs with probability at most $1/m$. **You can use this fact without proof**. The idea is simply to invoke the algorithm several times and rule by majority; See, e.g.,:
`https://homes.cs.washington.edu/~anuprao/pubs/CSE531Winter12/lecture7.pdf`
2. With a tiny modification, Grover's algorithm can find a 0 in a haystack of 1's (rather than a 1 in a haystack of 0's).
3. Recall that Grover's algorithm makes $O(\sqrt{N})$ queries (even if the number of 1's is unknown) and is correct with high probability (say, $2/3$), not with probability 1.
4. Say $A$ is an algorithm that is correct with high probability, and we want to run it $t$ times on $t$ different inputs. If we want all $t$ runs to be correct with high probability, we first need to reduce the error probability of $A$ to $O(1/t)$.