This exam is intended to take no more than 70 minutes. I am allowing 150 minutes to account for typing/recopying of solutions, technical problems, etc.

The exam is open-book and open-note — meaning:

- You may use all of this course's resources: the Mermin textbook, your class notes, your old homework, and the course web/Moodle site. You may not share any resources with any other person while you are taking the exam.
- You may cite material (definitions, theorems, examples, algorithms, etc.) from class, the assigned textbooks readings, and the assigned homework problems. You do not have to redevelop or reprove that material. On the other hand, you may not cite results that we have not studied.
- You may not consult any other books, papers, Internet sites, etc. You may use a computer for viewing the course web/Moodle site, typing and running Python programs of your own creation, typing up your answers, and e-mailing with me. If you want to use a computer for other purposes, then check with me first.
- You may not discuss the exam in any way — spoken, written, etc. — with anyone but me, until everyone has handed in the exam.

I will try to check my e-mail frequently during the exam period. Feel free to ask clarifying questions. If you cannot obtain clarification on a problem, then explain your interpretation of the problem in your solution. Never interpret a problem in a way that renders it trivial. Check your e-mail occasionally, in case I send out a correction.

Your solutions should be thorough, self-explanatory, neat, concise, and polished. Always show enough work and justification so that a typical classmate could understand your solutions. Correct answers without supporting work rarely earn full credit. You might want to work first on scratch paper and then recopy your solutions. Alternatively, you might want to type your solutions. If you cannot solve a problem, then write a brief summary of the approaches you've tried. Partial credit is often awarded. Present your solutions in the order assigned.

Good luck. :)

A. Suppose that $f : \{0,1\}^n \to \{0,1\}^n$. We have seen that the construction $F \cdot |\alpha\rangle|\beta\rangle = |\alpha\rangle|\beta \oplus f(\alpha)\rangle$ defines an $(n+n)$-qbit gate F by its effect on the standard basis (no matter what the specifics of f are). Which of the following variants also define $(n+n)$-qbit gates F, and which do not? Justify your answers.

1. $F \cdot |\alpha\rangle|\beta\rangle = |f(\beta) \oplus \alpha\rangle|\beta\rangle$.
2. $F \cdot |\alpha\rangle|\beta\rangle = |\alpha \oplus f(\alpha)\rangle|\beta\rangle$.
3. $F \cdot |\alpha\rangle|\beta\rangle = |\mathrm{rev}(\alpha)\rangle|\beta \oplus f(\alpha)\rangle$. Here, rev denotes the reversal operation on bit strings. For example, $\mathrm{rev}(010001) = 100010$.

B. A function $f : \{0,1\}^n \to \{0,1\}$ is said to be *fair* if $f(\alpha) = 0$ for 2^{n-1} inputs α and $f(\alpha) = 1$ for 2^{n-1} inputs α. Consider this problem: We are given an $f : \{0,1\}^n \to \{0,1\}$, and we are guaranteed that it is either constant or fair. Our job is to determine which it is — constant or fair.

What is the intersection of this problem with the Bernstein-Vazirani problem? That is, what is the largest problem that can be viewed as a special case of both problems?

C. In the context of Shor's algorithm, why does the continued fractions technique produce *at most one* qualifying rational approximation c/d to $b/2^n$? (In answering this question, you may assume results explicitly proven in lecture, homework, the Mermin textbook, etc. You may not assume other results.)

D. Why does Shor's algorithm require $2^n \geq m^2$? (I can think of four reasons. A thorough answer should mention all of them, but does not need to re-develop a bunch of argumentation around them.)