A.A. Excluding measurements, any n-qbit gate can be viewed as a(n) $\underline{2^n \times 2^n}$ $\underline{\text{unitary}}$ matrix.

A.B. If U and V are n-qbit gates, then $U \otimes V$ is a(n) $\underline{(2n)}$-qbit gate.

A.C. If U and V are n-qbit gates, then $U \cdot V$ is a(n) $\underline{n}$-qbit gate.

A.D. In the worst case, the Bernstein-Vazirani algorithm invokes the oracle f $\underline{1}$ time(s).

A.E. For any classical n-qbit state $|\alpha\rangle$, $H^{\otimes n}|\alpha\rangle = \underline{\displaystyle\sum_{\beta \in \{0,1\}^n} \frac{1}{2^{n/2}}(-1)^{\alpha \odot \beta}|\beta\rangle}$.

A.F. The construction $|\alpha\rangle|\beta\rangle \mapsto \underline{|\alpha\rangle|\beta \oplus f(\alpha)\rangle}$, for all $\alpha \in \{0,1\}^n$ and $\beta \in \{0,1\}^m$, turns any classical function $f : \{0,1\}^n \to \{0,1\}^m$ into its corresponding quantum gate.

B.A. [I'll omit the scratch work.] The quantum version of the classical AND gate is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

B.B. Mermin calls this gate Toffoli [as I mentioned in class] and CCNOT.

C.A. TRUE. [Any n-qbit state is a linear combination of the standard basis vectors. Each standard basis vector is a tensor product of 1-qbit classical states.]

C.B. TRUE. [It is $|0010\rangle$.]

C.C. TRUE. [Partial measurement collapses the state to some $|\alpha_\beta\rangle|\beta\rangle$, where $|\beta\rangle$ is classical and $|\alpha_\beta\rangle$ is the other part of the term in which $|\beta\rangle$ appeared.]

C.D. FALSE. [In $|-++++\rangle$, the first two qbits are unentangled from the last three. Therefore the partial measurement does not alter them. The post-measurement state is $|-+\alpha\beta\gamma\rangle$ for some $\alpha, \beta, \gamma \in \{0,1\}$.]

D. [I'll omit the drawing. There should be six wires entering on the left and six wires exiting on the right. There should be three CNOTs on the sixth wire, controlling by the first, fourth, and fifth wires.]

E.A. [I'll omit the drawing. On the left side should be wires for two registers. The input register (usually on top) is n qbits and the output register (bottom) is $n-1$ qbits. (They are all initially set to $|0\rangle$, although it's not clear that the question requires saying so.) The input register wires feed into a layer of Hadamard gates, while the output register wires do nothing. Then all of the wires feed into the quantum f gate. Then the output wires feed into a measurement gate, while the input wires do nothing. Then the input wires feed into another layer of Hadamard gates and then into a measurement gate.]

E.B. Each time the subroutine runs, it outputs a uniformly randomly generated bit string γ such that $\omega \odot \gamma = 0$. Once we have collected enough of these γ, we can solve the resulting linear system for ω. We need $n-1$ linearly independent γ, so the subroutine must run at least $n-1$ times. But sometimes it outputs redundant γ, so the number of runs is often larger than $n-1$. We proved in class that the expected number of runs is no larger than $2n$. We also proved in class that the expected number of runs is no larger than $n+1$.

F.A. Recall that each iteration of the loop rotates the state of the machine through an angle 2θ. As m increases, the angle θ increases, with two main effects. First, fewer iterations are needed to rotate the state vector close to $|\omega\rangle$. Thus the running time decreases. Second, we are not guaranteed to land as closely to $|\omega\rangle$. Thus the probability of finding a valid answer $|\omega_k\rangle$ decreases.

There are several other issues that could be raised. For example, when m is comparable to $\sqrt{2^n}$, Grover's algorithm has no speed advantage over simply guessing and checking. When m is around $\frac{1}{2}2^n$, k is in danger of rounding to 0. As θ gets far from 0, the approximation $\theta \approx \sin\theta$ that we used in the analysis degrades, so we have to change our analysis. If we want to discover all m solutions, then the expected number of successful runs of the algorithm is $\mathcal{O}(m^2)$, so it seems to increase as m increases.

F.B. Suppose that p and q are distinct large primes of about (and no more than) n bits each, and we're trying to factor the integer pq, which is about $2n$ bits in size. Set up a function f that takes an n-bit integer as input and returns whether that integer divides pq and is not 1. Thus f sends exactly $m = 2$ inputs to 1: p and q. Grover's algorithm applied to f finds p or q in time $\mathcal{O}(2^{n/2}) = \mathcal{O}(2^{m/4}) = \mathcal{O}(L_m[1, 1/4])$, where $m = 2n$ is the number of bits in pq. Unfortunately, there are classical factoring algorithms that find p or q in the much smaller time $\mathcal{O}(L[1/3, c])$.