1. For $g(x,y) = \frac{1}{2}x^T Px + q^T x + y^T(Ax-b)$, where $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^M$ what is $\nabla_x g(x,y)$?
   (a) $\nabla_x g(x,y) = q + Ax$
   (b) $\nabla_x g(x,y) = Px + q + A^T y$
   (c) $\nabla_x g(x,y) = q + A^T y$
   (d) $\nabla_x g(x,y) = (P-A)x$

   Solution: b

2. For $g(x,y) = \frac{1}{2}x^T Px + q^T x + (Ax-b)^T y$, where $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^M$ what is $\nabla_x g(x,y)$?
   (a) $\nabla_x g(x,y) = q + Ax$
   (b) $\nabla_x g(x,y) = Px + q + A^T y$
   (c) $\nabla_x g(x,y) = q + A^T y$
   (d) $\nabla_x g(x,y) = (P-A)x$

   Solution: b, changing $y^T(Ax-b)$ to $(Ax-b)^T y$ does not change the derivatives (since It's a scalar).

3. For $g(x,y) = \frac{1}{2}x^T Px + q^T x + (Ax-b)^T y$, where $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^M$ what is $\nabla_y g(x,y)$?
   (a) $\nabla_y g(x,y) = A^T x$
   (b) $\nabla_y g(x,y) = Ax - b$

   Solution: b

4. For $g(x,y) = \frac{1}{2}x^T Px + q^T x + y^T(Ax-b)$, where $x \in \mathbb{R}^N$ and $y \in \mathbb{R}^M$ what is $\nabla_y g(x,y)$?
   (a) $\nabla_y g(x,y) = A^T x$
   (b) $\nabla_y g(x,y) = Ax - b$

   Solution: b, same reasoning as 2.

5. For $J(x) = (x - x_{ref})^T Q(x - x_{ref})$, what is $\nabla_x J(x)$?
   (a) $\nabla_x J(x) = Qx - Qx_{ref}$
   (b) $\nabla_x J(x) = 2Q(x - x_{ref})$
   (c) $\nabla_x J(x) = Q(x - x_{ref})$

   Solution: b, we forgot to include that $Q \in S$ (Q is symmetric).

6. Both iLQR and DDP solve the same class of trajectory optimization problem.
   (a) true
   (b) false

   Solution: True, they both solve unconstrained trajectory optimization problems. (unconstrained meaning the only constraints present are the dynamics constraints, they cannot handle general constraints like control and state limits.

7. DDP is simply the Gauss-Newton version of iLQR (which is full Newton).
   (a) true
   (b) false

   Solution: False, DDP is computing full-newton steps (it's calculating second derivatives of the constraints), and iLQR is Gauss-Newton since it linearizes the constraints before forming the quadratic approximation of the cost to go function.

8. With infinite precision, iLQR with a quadratic cost function and nonlinear dynamics would not need regularization during the backwards pass to ensure positive semi-definiteness of the cost-to-go hessian P.

   (a) true

   (b) false

   Solution: True, since iLQR is Gauss-Newton, given a quadratic convex cost function, there should not require any regularization to ensure positive definiteness of P. In reality, a little regularization can sometimes help numerical robustness of the solver.

9. With infinite precision, DDP with a quadratic cost function and nonlinear dynamics would not need regularization during the backwards pass to ensure positive semi-definiteness of the cost-to-go hessian P.

   (a) true

   (b) false

   Solution: False, DDP is Full-Newton, so there is a potential for negative eigenvalues to appear in P. This exact same logic applies to any generic Full-Newton vs Gauss-Newton debate.

10. In iLQR/DDP, you can initialize the solver with a dynamically infeasible initial guess.

    (a) true

    (b) false

    Solution: False, you can only provide an initial set of controls $u_{1:N-1}$ to iLQR/DDP. This means the initial guess for the solver will always be the result of a dynamics rollout, and will therefore always be dynamically feasible.