1. Consider the following proof
    a. Property of matrix Q
        i. $equation\ u^T Qu \geq 0\ prove\ that\ the\ matrix\ Q\ is\ positive\ semi-definite\ for\ all\ u \in \Re^n$
            1. $more\ specifically, it\ is\ symmetric\ positive\ semi-definite\ (SPSD)$
    b. Property of matrix Q mean for the standard QP problem
        i. $The\ above\ property\ prove\ that\ QP\ is\ a\ convex\ function$
    c. Usefulness of (b)
        i. Since we are looking for a u^* that minimize the $f(x)$, and since the function is convex, by finding the absolute minima will make you find the answer.
2. Given the standard QP problem, explain what each component represents.
    a. u
        i. this is the part that has bias and weights that QP solver is trying to optimize.
        ii. this is useful since this will give us the hyperplane.
    b. Q
        i. $q \times q\ matrix$ that represents coefficients of quadratic term.
        ii. Useful since this is the one that changes the optimal value u that we are looking for
    c. p
        i. $q \times 1\ vector$ that represents coefficients of linear term.
        ii. Same as Q, this can change the optimal value u that we are looking for since it is the coefficients
    d. A
        i. $q \times 1\ vector$ that specifies the linear inequality constraints.
        ii. We use this vector, constructed from this equation $y_n(w^T x_n + b) \geq 1$, where A is equivalent to this part $y_n(w^T x_n + b)$ that has to be solved with c vector below in order to get u
    e. c
        i. $q \times 1\ vector$ which normally appear as 1 for each element.
        ii. Similar to A vector, it represents 1 from the equation above, looks like this $[1, 1, 1, \dots 1]$ and used to solve for u with A vector.
3. Consider the dataset, manually solve the optimal hyperplane optimization problem
    a. $\min \frac{1}{2} w^t w \quad subject\ to: y_n(w^T x_n + b) \geq 1$
    b. $X = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ -2 & 0 \end{bmatrix} \quad y = \begin{bmatrix} -1 \\ -1 \\ +1 \end{bmatrix} \quad \begin{cases} 1: & -b \geq 1 \\ 2: & -(-w_2 + b) \geq 1 \\ 3: & -2w_1 + b \geq 1 \end{cases}$
    c. Matrix Notation

i. $\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ -2 & 0 \end{bmatrix} w + b * y \geq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

ii. $\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & -1 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

iii. We know that $b = -1\ from - b \geq 1$

1. $-(-w_2 + b) \geq 1 \ => w_2 - b \geq 1 = \ w_2 + 1 \geq 1 \ => \ w_2 \geq 0$
   a. $w_2 = 0$
2. $-2w_1 + b \geq 1 => \ -2w_1 - 1 \geq 1 => -2w_1 \geq 2 => w_1 \leq -1$
   a. $w_1 = -1$

d. $b = -1, w_1 = -1, w_2 = 0$

4. SVM using python
   a. Use CVXOPT to verify the toy dataset
      i.

```
     pcost       dcost       gap    pres   dres
 0:  3.2653e-01  1.9592e+00  6e+00  2e+00  4e+00
 1:  1.5796e+00  8.5663e-01  7e-01  2e-16  2e-15
 2:  1.0195e+00  9.9227e-01  3e-02  2e-16  2e-15
 3:  1.0002e+00  9.9992e-01  3e-04  2e-16  1e-15
 4:  1.0000e+00  1.0000e+00  3e-06  3e-16  3e-15
 5:  1.0000e+00  1.0000e+00  3e-08  0e+00  1e-15
Optimal solution found.
[-1.00e+00]
[ 1.00e+00]
[-1.00e+00]
```

         1. Python code is submitted on Mimir

   b. From what I tested, increasing dimension cause the runtime cost to increase faster than increasing sample size. These are the two scenario that my computer started to slow down and took a minute to calculate.
      i. Sample_size: 10 Dimension: 20000
         1. If you have less than 5000 dimensions, it will calculate the optimal solution quickly.
      ii. Sample_size: 10,000,000 Dimension: 2
         1. If you have less than 1,000,000 samples, it will calculate the optimal solution quickly.