Python code

```python
import numpy as np
import cvxpy as cp

print("1. Use cvxpy library")
x1 = cp.Variable()
x2 = cp.Variable()

constraint = [ x1 <= 1,
               x2 <= 1,
               -x1 <= 1,
               -x2 <= 1]

obj = cp.Minimize(x1**2 + 2*x2**2 + x1*x2 + 5*x1 + 3*x2)

prob = cp.Problem(obj, constraint)
print("=="*20) # 만약 infeasible 이거나, 문제가 있을 경우 여기서 알 수 있음.
if prob.status=='optimal':
    print(f"status: {prob.status}")
    print("optimal value", prob.value)
    print("optimal var", x1.value, x2.value)
else:
    print(f"Wrong status: {prob.status}")
print("=="*20)
print("\n\n")

print("2. Use crude method, increasing dual variable little by little every time")

Q = np.array([[2, 1],[1, 4]]).astype('float64')
c = np.array([[5], [3]]).astype('float64').reshape(2, 1)
A = np.array([[1, 0], [-1, 0], [0, 1], [0, -1]]).reshape(4, 2).astype('float64')
b = np.array([[1], [1], [1], [1]]).astype('float64')

def min(y, Q, c):
#  return 0.5*((y.T).dot(Q)).dot(y) + (c.T).dot(y)
  return y1**2 + 2*y2**2 + y1*y2 + 5*y1 + 3*y2
y1 = -1
y2 = -1
y = np.array([y1, y2]).reshape(2, 1).astype('float32')

base = min(y, Q, c)
result = base
save_y1 = y1
save_y2 = y2

for i in range(0,2001):
  y1 += 0.001
  y2 = -1
  for j in range(0, 2001):
    y2 += 0.001
    update = min(y, Q, c)
    if(base > update):
      result = update
      base = update
      save_y1 = y1
```

```
        save_y2 = y2

print("=="*20)
print("result:",round(result,3))
print("y1 =", round(save_y1,3))
print("y2 =" , round(save_y2,3))
print("=="*20)

# y1 ~ -1, y2 = -0.5 로 계산을 다시하면
y1 = -1
y2 = -0.5
print("minimum of objectiv function is : ", min(y, Q, c))
```

Code 설명

가장 먼저 quadratic problem 을 풀 때 사용하는 library 인 cvxpy 를 불러왔다. cvxpy 를 cp 로 불러온 뒤, 행렬 연산을 편하게 하기 위해 numpy 를 불러왔다. Line5, 6 에서 x1 과 x2 에 cp.variable() 함수로 변수를 지정해주고, line13 에서 quadratic problem 의 objective function 을 정의해주었다. 바로 이어 constraint 를 지정해준 뒤 objective function 과 constraint 를 묶어 cp.Problem 의 인자로 전달하여 quadratic problem 을 정의한다. 하지만, 이렇게 했을 때 함수의 출력으로 아래와 같이 나왔다.

```
========================================
Wrong status: None
========================================
```

행렬을 다시 정의해주고 objective function 도 모두 풀어 방정식의 형태로 정의해 주었는데도 동일한 결과가 출력되었다.

해서 두번째 방법으로 단순하게 x1, x2 두개의 변수를 조금씩 증가시키면서 최소값을 찾는 방법을 사용하였다. Colab 환경에서 진행하였기에 x1, x2 대신 y1, y2 로 바꾸었고, line 33 에서 objective function 을 직접 정의하였다.
2 개의 변수 모두 -1 부터 1 까지 총 2 의 증가량을 0.001 만큼 증가시켰고(line 45~50), 증가할 때마다 objective function 에 값을 대입해 이전까지의 최솟값과 비교하여 더 작은 값과 그에 해당하는 y1, y2 값을 저장하였다.
그 결과 아래와 같은 결과가 나왔다.

```
result: -4.497
y1 = -0.999
y2 = -0.5
```