2. (We are only counting multiplications + additions)

a) From week 5 slides, we know that the cost of computing the inverse of a $n \times n$ matrix is n^3 flops if we take advantage of the particular form of the right-hand side vectors eg. Then, matrix multiplication of $n \times n \cdot n \times k$ matrices takes $n \times n \times k$ flops $= kn^2$ flops. Therefore, the total is $n^3 + kn^2$ flops.

b) From week 5 slides, we know the cost of LU factorization on A is $\frac{n^3}{3}$ flops, and for each column of B, the forward and backword substitutions cost $\frac{n^2}{2} \times 2$ in total, and there are k columns. Thus, the total is $\frac{n^3}{3} + kn^2$ flops.

c) Thus, algorithm II is more efficient, since it only takes $\frac{n^3}{3} + kn^2$ flops, while algorithm I takes $n^3 + kn^2$ flops, the algorithm II can save about $\frac{2n^3}{3}$ flops. Thus it is more efficient.

3.

```
q3(A, B, C, b):
    M = sla.solve(B, 2A+1)
    N = sla.solve(C, b)
    x = np.dot(M, N+np.dot(A, b))
    return x
```