```
Input: vector x, vector y
Output: coefitiens, Matrix A

begin cuadraticSpline

    if x or y has duplicates:
        return error
    end if

    if lenght of x is not equals to lenght of y:
        return error
    end if

    set n = lenght of x
    set m = (n - 1) * 3
    set A = matrix[m][m]
    set B = vector[m]
    set A[0][0] = x[0] ^2
    set A[0][1] = x[0]
    set A[0][2] = 1
    set B[0] = y[0]
    #interpolation conditions
    For i = 0,..., n      1
        set A[i+1][3*(i+1)-3] = math.pow(x[i+1], 2)
        set A[i+1][3*(i+1)-2] = x[i+1]
        set A[i+1][3*(i+1)-1] = 1
        set B[i+1] = y[i+1]
    end for
    #continuity conditions
    For i = 1,..., n      1
        set  A[n-1+i][3*i-3] = math.pow(x[i], 2)
        set  A[n-1+i][3*i-2] = x[i]
        set  A[n-1+i][3*i-1] = 1
        set  A[n-1+i][3*i] = -math.pow(x[i], 2)
        set  A[n-1+i][3*i+1] = -x[i]
        set  A[n-1+i][3*i+2] = -1
        set  B[n-1+i] = 0
    end for
    #softness condition
    for i = 1,..., n      1
        set A[2*n-3+i][3*i-3] = 2 * x[i]
        set   A[2*n-3+i][3*i-2] = 1
        set  A[2*n-3+i][3*i-1] = 0
        set  A[2*n-3+i][3*i] = -2 * x[i]
        set  A[2*n-3+i][3*i+1] = -1
        set  A[2*n-3+i][3*i+2] = 0
```

```
            set   B[2*n-3+i] = 0
        end for
        set   A[m-1][0] = 2
        set   B[m-1] = 0
        x = solveSystem(A, B)
        return x, A

end cuadraticSpline
```