In [18]:

```python
import numpy as np
A = np.array([[25,0,1],[20,1,2],[40,1,6]])
b = np.array([[110],[110],[210]])

# To see rank, use:
# np.linalg.matrix_rank(A)

# To invert a matrix, use:
# np.linalg.inv(A)

# To multiply matrices in Python 3, use:
# A@B
```

In [19]:

```python
# Problem 2a.)

np.linalg.inv(A)@b

# The array does not agree with the known values.
```

Out[19]:

```
array([[ 4.25],
       [17.5 ],
       [ 3.75]])
```

In [54]:

```python
import numpy as np
A = np.array([[25,15,10,0,1],[20,12,8,1,2],[40,30,10,1,6], [30,15,15,0,3], [35,20,15,2,4
b = np.array([[104],[97],[193],[132],[174]])

print(np.linalg.matrix_rank(A))

# Note: you can use np.hstack() to concatinate vectors, for example np.hstack((A,b))

# Note: you can select all the columns, except the first of a matrix A as:  A[:,1:]
```

```
4
```

In [56]:

```python
# Problem 2b.)

Awithb = np.hstack((A,b))
print(f"Rank of a with b appended: {np.linalg.matrix_rank(Awithb)}")

# i. rank(A) = 4
#    rank(A | b) = 4
#    because rank(A) = rank(A | b), b does lie within the span of A.
#    So at least one exact solution does exist.

# ii. rank(A) = 4
#     dim(x) = 5
#     More items in x than linearly independent equations.
#     Underdetermined system, infinitely many solutions.

# iii.

Awithout1stcol = A[:,1:]
print(f"new rank without 1st col: {np.linalg.matrix_rank(Awithout1stcol)}")
# So this new matrix without the 1st column is rank 4
# but the x-vector is dimension 4 now.

np.linalg.matrix_rank(np.hstack((Awithout1stcol,b)))
# Before: 5x5, 5x1 = 5x1
# After removing 1st col: 5x4, 4x1 = 5x1

# So there is now a single unique solution now
# Cannot simply multiply A^-1 to both sides since Awithout1stcol is nonsquare
# Use transpose method
```

```python
Awithout1stcol_transpose = np.matrix.transpose(Awithout1stcol)
inverse = np.linalg.inv(Awithout1stcol_transpose @ Awithout1stcol)
x =  inverse @ Awithout1stcol_transpose @ b
print(x)

# The squared error will be zero since b is within span(A).
print(b-Awithout1stcol@x)

# A bunch of floating point errors if I had to guess
```

```
Rank of a with b appended: 4
new rank without 1st col: 4
[[4.]
 [4.]
 [9.]
 [4.]]
[[-1.70530257e-13]
 [-1.56319402e-13]
 [ 0.00000000e+00]
 [ 8.52651283e-14]
 [-2.27373675e-13]]
```

In [ ]:

In [ ]: