In [1]:
```
import numpy as np
```

## Question 1

### a)

In [2]:
```
road = [1, 1, 0, 0, 0]
settlement = [1, 1, 1, 1, 0]
city = [0, 0, 2, 0, 3]
development_card = [0, 0, 1, 1, 1]
```

In [3]:
```
building_costs = np.matrix([road, settlement, city,
                            development_card])

print(building_costs)
```

```
[[1 1 0 0 0]
 [1 1 1 1 0]
 [0 0 2 0 3]
 [0 0 1 1 1]]
```

Rows represent things we can build (road, settlement, city, and development card), and columns represent material types that we might need to build them. The order of columns from left to right is: [brick, wood, wheat, sheep, ore]

### b)

In [4]:
```
costs = np.array([2, 1, 5, 3, 8])
```

In [5]:
```
building_costs @ costs
```

Out[5]:
```
matrix([[ 3, 11, 34, 16]])
```

Road: 3 , Settlement: 11 , City: 34

### c)

In [6]:
```
order = np.array([6, 2, 1, 0])
```

In [7]:
```
materials_needed = building_costs.T @ order
print(materials_needed)
```

```
[[8 8 4 2 3]]
```

### d)

In [8]:
```
materials_needed @ costs
```

Out[8]:
```
matrix([[74]])
```

## Question 2

### a)

In [9]:
```
X = np.matrix([[8, 0, 1, 1],
               [9, 2, 9, 4],
               [1, 5, 9, 9],
               [9, 9, 4, 7],
               [6, 9, 8, 9]])
```

In [10]:
```
y = np.array([0, 0, 0, 1, 0])
```

In [11]:
```
y.T @ X
```

Out[11]:
```
matrix([[9, 9, 4, 7]])
```

### b)

A vector with all zeroes except a 1 at the kth position.

In [12]:
```
def kthrow(k):
    y = np.zeros(5)
    y[k - 1] = 1
    return y
```

In [13]:
```
for i in (range(1, 6)):
    print(f'k = {i}:', kthrow(i))
```

```
k = 1: [1. 0. 0. 0. 0.]
k = 2: [0. 1. 0. 0. 0.]
k = 3: [0. 0. 1. 0. 0.]
k = 4: [0. 0. 0. 1. 0.]
k = 5: [0. 0. 0. 0. 1.]
```

In [14]:
```
for i in range(1, 6):
    print(f'Row {i}', kthrow(i) @ X)
```

```
Row 1 [[8. 0. 1. 1.]]
Row 2 [[9. 2. 9. 4.]]
Row 3 [[1. 5. 9. 9.]]
Row 4 [[9. 9. 4. 7.]]
Row 5 [[6. 9. 8. 9.]]
```

### c)

Vector of size 5 with all zeros except a in kth position and b in jth position.

```
In [15]: def cons_vector(a, b, k, j, n=5):
             y = np.zeros(n)
             y[k - 1] = a
             y[j - 1] = b
             return y
```

```
In [16]: print('First row times 2 + second row times 3:\n',
               cons_vector(2, 3, 1, 2))
```

```
First row times 2 + second row times 3:
 [2. 3. 0. 0. 0.]
```

```
In [17]: print(cons_vector(2, 3, 1, 2).T @ X)
```

```
[[43.  6. 29. 14.]]
```

**d) & e)**

All zeros except a 1 at the kth position

```
In [18]: w = np.array([0, 0, 1, 0])
```

```
In [19]: X @ w
```

```
Out[19]: matrix([[1, 9, 9, 4, 8]])
```

**f)**

Vector of size 4 with all zeros except a in kth position and b in jth position.

```
In [20]: w = cons_vector(2, 3, 1, 2, 4)
```

```
In [21]: X @ w
```

```
Out[21]: matrix([[16., 24., 17., 45., 39.]])
```

## Question 3

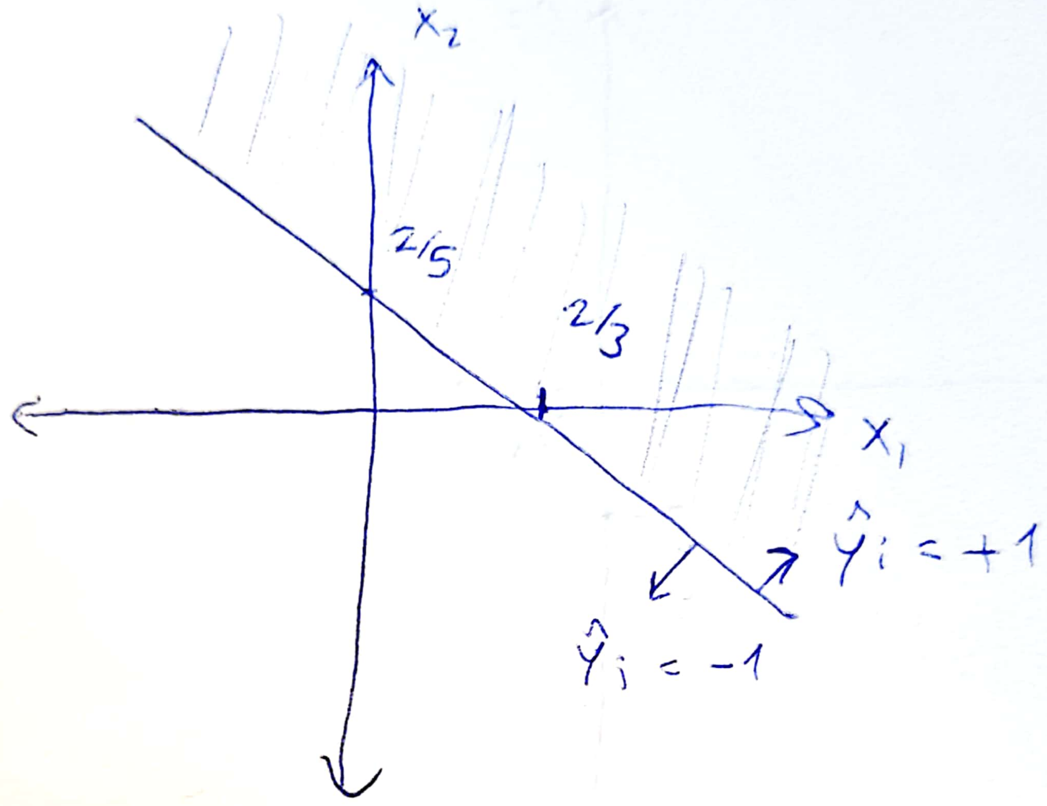
All columns and rows are linearly dependent. e.g. dividing row 1 with 3 and row 3 with row results in a matrix with identical rows.

Similarly, dividing column 1 with 5, 2 with 9, 3 with 2, 4 with 6 results in a matrix with identical columns.

## Question 4

## Question 4

Question 4



$$3x_{i,1} + 5x_{i,2} + (-2) = 0$$

## Question 5

a) $y = w_1 + w_2 z_i + w_3 z_i^2 \cdots w_{d+1} z_i^d$

b)
$$n\left\{\overbrace{\begin{bmatrix}1 & z_1 & z_1^2 & z_1^3 & \cdots & z_1^d\\ 1 & z_2 & z_2^2 & z_2^3 & \cdots & z_2^d\\ 1 & z_3 & z_3^2 & z_3^3 & \cdots & z_3^d\\ \vdots & \vdots & \vdots & \vdots & & \vdots\\ 1 & z_n & z_n^2 & z_n^3 & \cdots & z_n^d\end{bmatrix}}^{d+1}\right. \begin{bmatrix}w_1\\ w_2\\ \vdots\\ w_{d+1}\end{bmatrix} = \left.\begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_n\end{bmatrix}\right\}n$$

includes the intercept

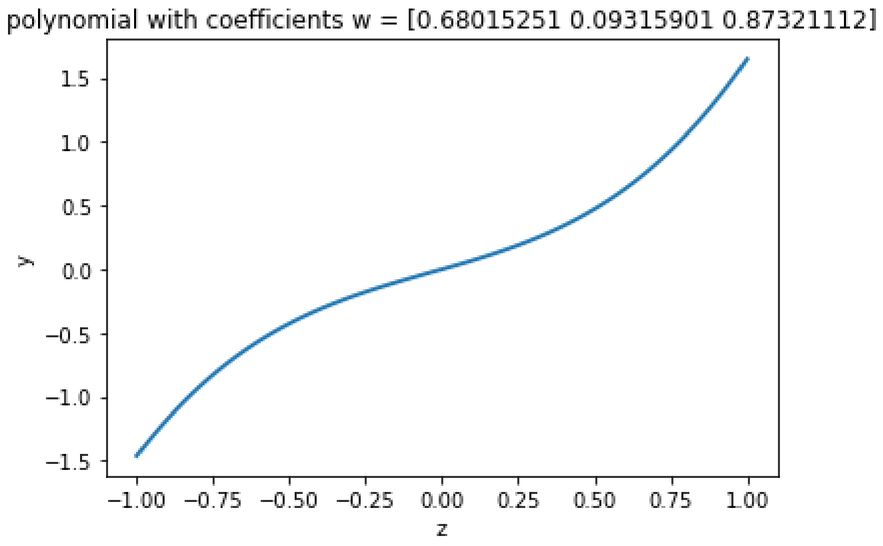
**c)**

In [105]:
```
import scipy.io as sio
import matplotlib.pyplot as plt
# n = number of points
# z = points where polynomial is evaluated
# p = array to store the values of the interpolated polynomials
n = 100
z = np.linspace(-1, 1 , n)
d = 3 # degree
w = np.random.rand(d)
X = np.zeros((n , d))
```

In [106]:
```
# generate X- matrix
for i in range(1, d + 1):
    X[:, i - 1] = z ** i

# evaluate polynomial at all points z, and store the result in p
p = w @ X.T

# plot the datapoints and the best -fit polynomials
plt.plot(z, p, linewidth=2)
plt.xlabel('z')
plt.ylabel('y')
plt.title('polynomial with coefficients w = %s' % w )
plt.show()
```


polynomial with coefficients w = [0.68015251 0.09315901 0.87321112]
y
z