IMPORTING LIBRARIES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

GENERATING 200 POINTS UNIFORMLY SPACED IN THE INTERVAL [-1,1]

In [2]:
```
step = (2/200)
X = np.arange(-1,1,step)
X
```

Out[2]:
```
array([-1.0000000e+00, -9.9000000e-01, -9.8000000e-01, -9.7000000e-01,
       -9.6000000e-01, -9.5000000e-01, -9.4000000e-01, -9.3000000e-01,
       -9.2000000e-01, -9.1000000e-01, -9.0000000e-01, -8.9000000e-01,
       -8.8000000e-01, -8.7000000e-01, -8.6000000e-01, -8.5000000e-01,
       -8.4000000e-01, -8.3000000e-01, -8.2000000e-01, -8.1000000e-01,
       -8.0000000e-01, -7.9000000e-01, -7.8000000e-01, -7.7000000e-01,
       -7.6000000e-01, -7.5000000e-01, -7.4000000e-01, -7.3000000e-01,
       -7.2000000e-01, -7.1000000e-01, -7.0000000e-01, -6.9000000e-01,
       -6.8000000e-01, -6.7000000e-01, -6.6000000e-01, -6.5000000e-01,
       -6.4000000e-01, -6.3000000e-01, -6.2000000e-01, -6.1000000e-01,
       -6.0000000e-01, -5.9000000e-01, -5.8000000e-01, -5.7000000e-01,
       -5.6000000e-01, -5.5000000e-01, -5.4000000e-01, -5.3000000e-01,
       -5.2000000e-01, -5.1000000e-01, -5.0000000e-01, -4.9000000e-01,
       -4.8000000e-01, -4.7000000e-01, -4.6000000e-01, -4.5000000e-01,
       -4.4000000e-01, -4.3000000e-01, -4.2000000e-01, -4.1000000e-01,
       -4.0000000e-01, -3.9000000e-01, -3.8000000e-01, -3.7000000e-01,
       -3.6000000e-01, -3.5000000e-01, -3.4000000e-01, -3.3000000e-01,
       -3.2000000e-01, -3.1000000e-01, -3.0000000e-01, -2.9000000e-01,
       -2.8000000e-01, -2.7000000e-01, -2.6000000e-01, -2.5000000e-01,
       -2.4000000e-01, -2.3000000e-01, -2.2000000e-01, -2.1000000e-01,
       -2.0000000e-01, -1.9000000e-01, -1.8000000e-01, -1.7000000e-01,
       -1.6000000e-01, -1.5000000e-01, -1.4000000e-01, -1.3000000e-01,
       -1.2000000e-01, -1.1000000e-01, -1.0000000e-01, -9.0000000e-02,
       -8.0000000e-02, -7.0000000e-02, -6.0000000e-02, -5.0000000e-02,
       -4.0000000e-02, -3.0000000e-02, -2.0000000e-02, -1.0000000e-02,
        8.8817842e-16,  1.0000000e-02,  2.0000000e-02,  3.0000000e-02,
        4.0000000e-02,  5.0000000e-02,  6.0000000e-02,  7.0000000e-02,
        8.0000000e-02,  9.0000000e-02,  1.0000000e-01,  1.1000000e-01,
        1.2000000e-01,  1.3000000e-01,  1.4000000e-01,  1.5000000e-01,
        1.6000000e-01,  1.7000000e-01,  1.8000000e-01,  1.9000000e-01,
        2.0000000e-01,  2.1000000e-01,  2.2000000e-01,  2.3000000e-01,
        2.4000000e-01,  2.5000000e-01,  2.6000000e-01,  2.7000000e-01,
        2.8000000e-01,  2.9000000e-01,  3.0000000e-01,  3.1000000e-01,
        3.2000000e-01,  3.3000000e-01,  3.4000000e-01,  3.5000000e-01,
        3.6000000e-01,  3.7000000e-01,  3.8000000e-01,  3.9000000e-01,
        4.0000000e-01,  4.1000000e-01,  4.2000000e-01,  4.3000000e-01,
        4.4000000e-01,  4.5000000e-01,  4.6000000e-01,  4.7000000e-01,
        4.8000000e-01,  4.9000000e-01,  5.0000000e-01,  5.1000000e-01,
        5.2000000e-01,  5.3000000e-01,  5.4000000e-01,  5.5000000e-01,
        5.6000000e-01,  5.7000000e-01,  5.8000000e-01,  5.9000000e-01,
        6.0000000e-01,  6.1000000e-01,  6.2000000e-01,  6.3000000e-01,
        6.4000000e-01,  6.5000000e-01,  6.6000000e-01,  6.7000000e-01,
```

```
6.8000000e-01,  6.9000000e-01,  7.0000000e-01,  7.1000000e-01,
7.2000000e-01,  7.3000000e-01,  7.4000000e-01,  7.5000000e-01,
7.6000000e-01,  7.7000000e-01,  7.8000000e-01,  7.9000000e-01,
8.0000000e-01,  8.1000000e-01,  8.2000000e-01,  8.3000000e-01,
8.4000000e-01,  8.5000000e-01,  8.6000000e-01,  8.7000000e-01,
8.8000000e-01,  8.9000000e-01,  9.0000000e-01,  9.1000000e-01,
9.2000000e-01,  9.3000000e-01,  9.4000000e-01,  9.5000000e-01,
9.6000000e-01,  9.7000000e-01,  9.8000000e-01,  9.9000000e-01])
```

CREATING Y AS GIVEN IN THE QUESTION

In [3]:

```
Y=[]
for i in X:
    if((i>=-0.5 and i<0.1) or (i>=0.5)):
        Y.append(1)
    else:
        Y.append(-1)
Y=np.array(Y)
Y
```

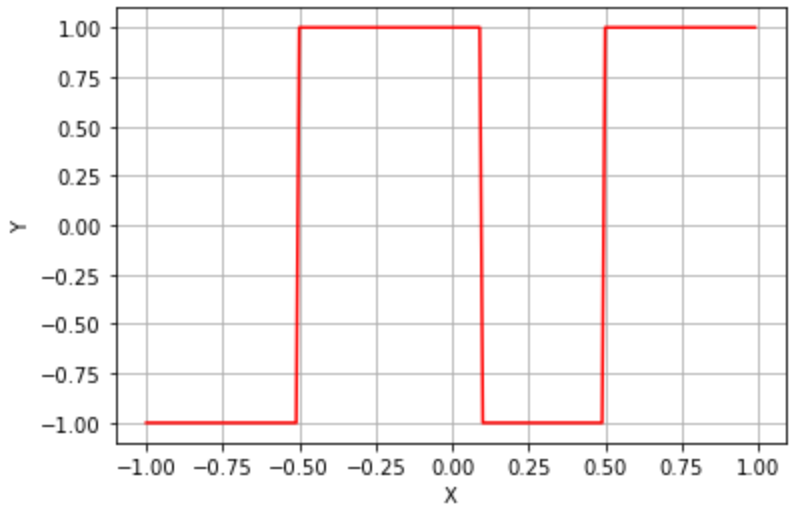
Out[3]:

```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1])
```

PLOTTING X & Y

In [4]:

```
plt.plot(X,Y,color='red')
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.show()
```

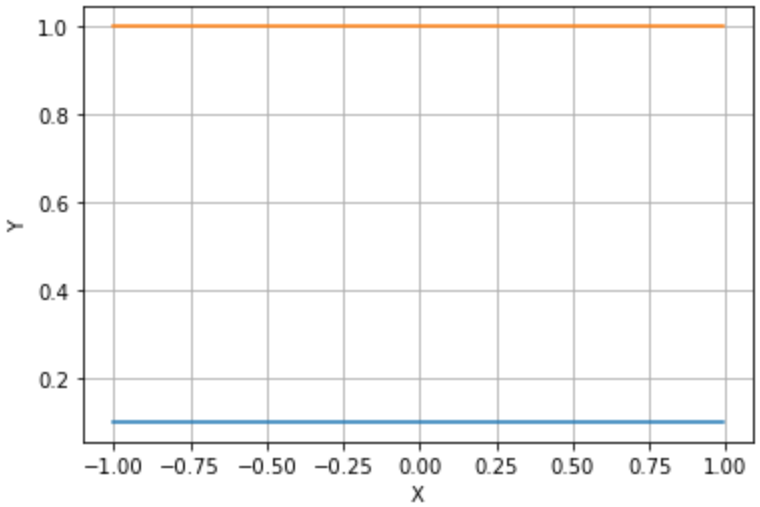


Answer Of Q2.b

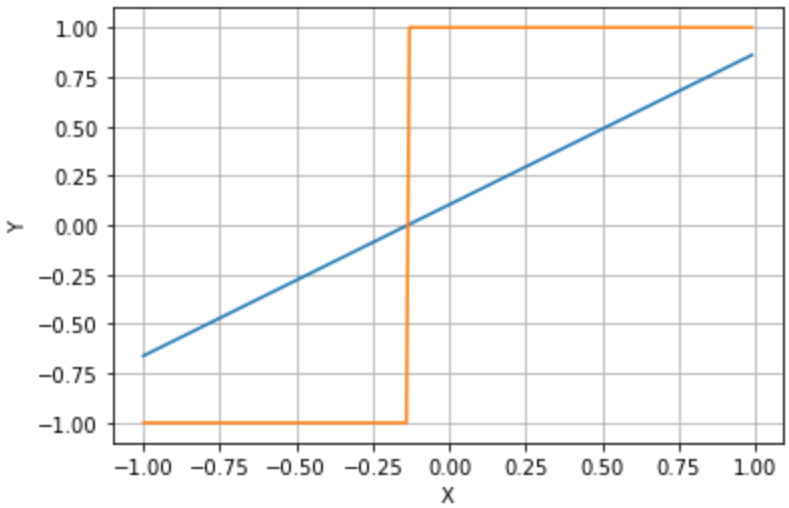
In [5]:

```
errors=[]
for degree in range(0,9):
    poly=PolynomialFeatures(degree)
    Xdf=pd.DataFrame(X)
    Xpoly=poly.fit_transform(Xdf)
    lr.fit(Xpoly,Y)
    print("Degree : ",end="")
    print(degree)
    print("Train Accuracy : ",end="")
    score = lr.score(Xpoly,Y)
    print(score)
    Ypred=lr.predict(Xpoly)
    if(degree==3):
        deg3pol=Ypred
    RMS=(1/200)*np.sum(np.square(Y-Ypred))
    print("Error(RMS) : ",end="")
    print(RMS)
    errors.append(RMS)
    plt.plot(X,Ypred)
    plt.plot(X,np.sign(Ypred))
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.grid()
    plt.show()
```

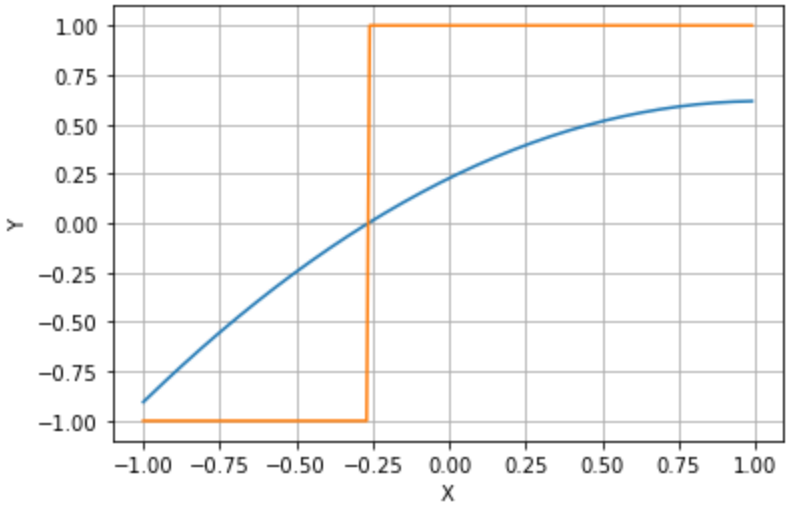
```
Degree : 0
Train Accuracy : 0.0
Error(RMS) : 0.9900000000000003
```



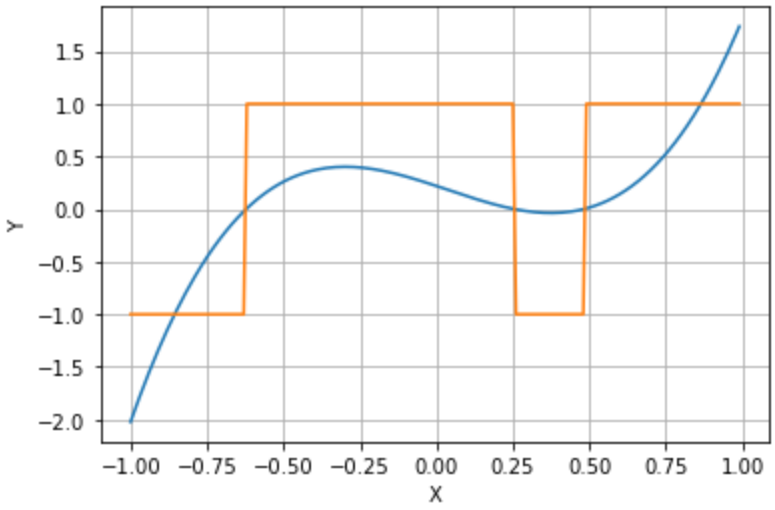

```
Degree : 1
Train Accuracy : 0.19705038080497495
Error(RMS) : 0.7949201230030751
```



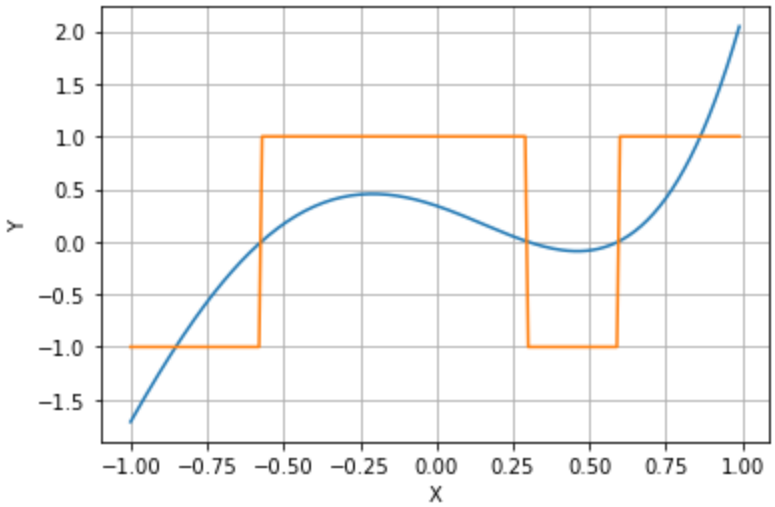
```
Degree : 2
Train Accuracy : 0.20942692784241324
Error(RMS) : 0.7826673414360111
```



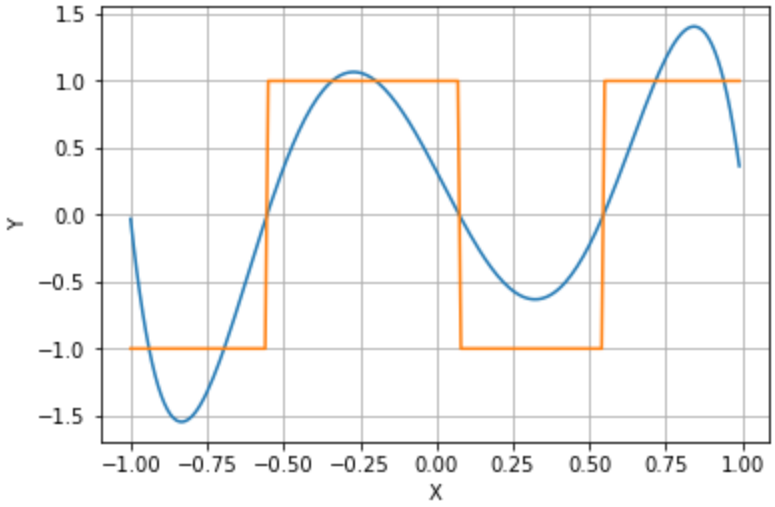

```
Degree : 3
Train Accuracy : 0.40069842880320705
Error(RMS) : 0.5933085554848252
```



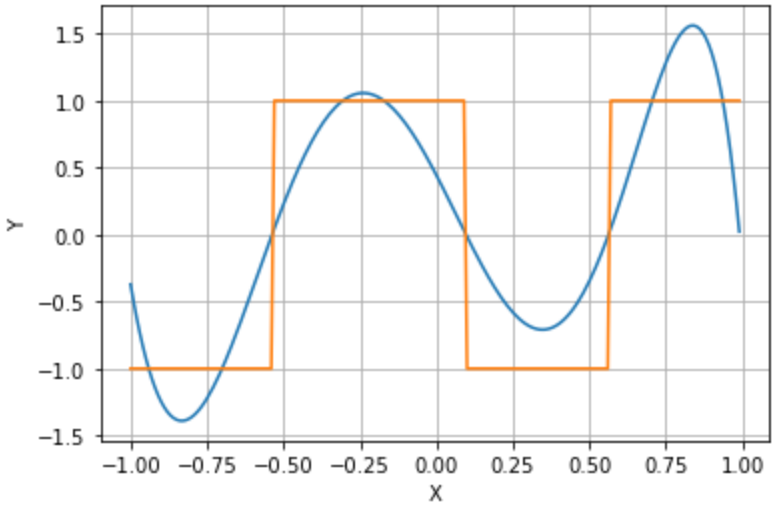

```
Degree : 4
Train Accuracy : 0.4126551059785555
Error(RMS) : 0.5814714450812302
```



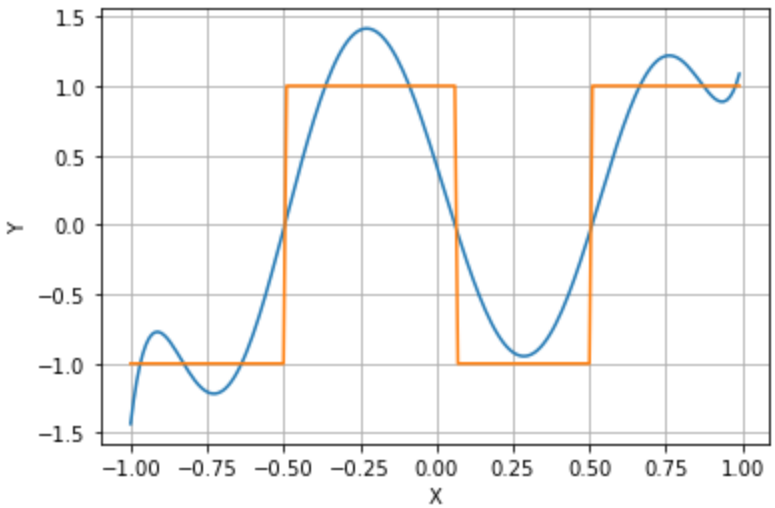

```
Degree : 5
Train Accuracy : 0.7141288136135671
Error(RMS) : 0.2830124745225687
```



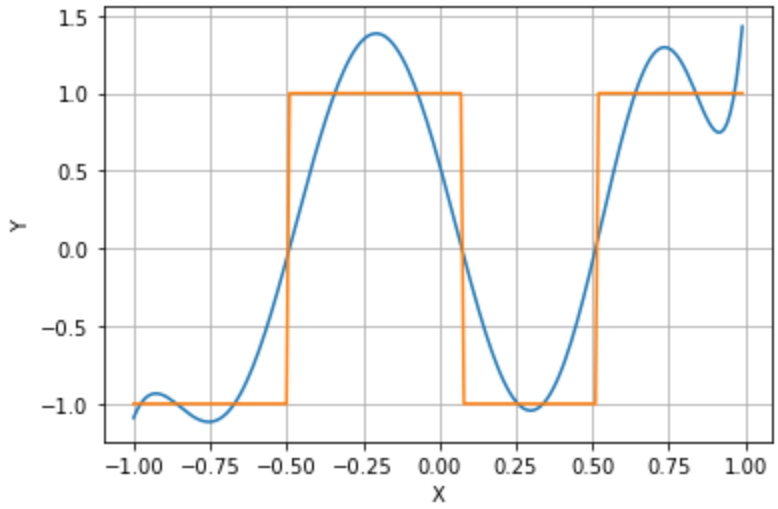

```
Degree : 6
Train Accuracy : 0.7252590694544849
Error(RMS) : 0.27199352124006004
```




```
Degree : 7
Train Accuracy : 0.8259180227766211
Error(RMS) : 0.1723411574511452
```



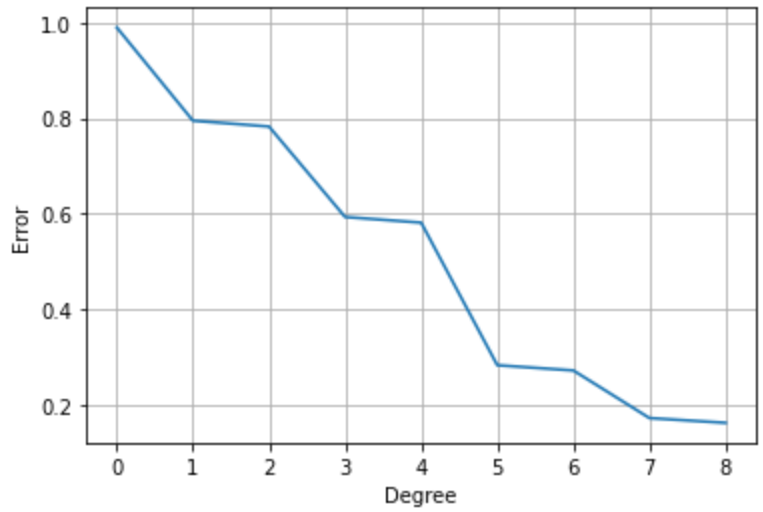

```
Degree : 8
Train Accuracy : 0.8359627493186512
Error(RMS) : 0.1623968781745354
```



Answer Of Q2.a

PLOTTING ERROR WITH RESPECT TO DEGREE

In [6]:
```
plt.plot(errors)
plt.xlabel("Degree")
plt.ylabel("Error")
plt.grid()
plt.show()
```



Yes,the error rate decreases when we increase the degree of the polynomial as shown in the figure above.

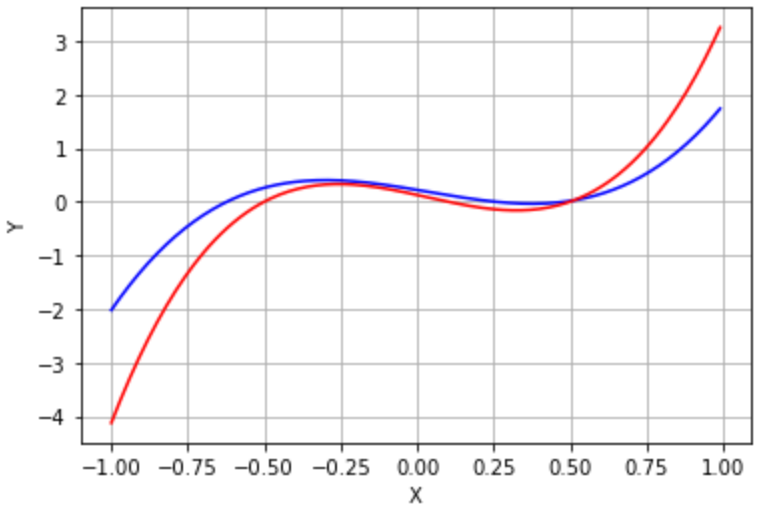
Answer Of Q2.c

In [7]:
```
def Q2_c_fun(x,c):
    ans=[]
    for i in x:
        ans.append(c*(i+0.5)*(i-0.1)*(i-0.5))
    return ans
```

In [8]:
```
Cubic_Polynomial = Q2_c_fun(X,5)
```

Plotting the polynomials

In [9]:
```
plt.plot(X,deg3pol,color="blue")
plt.plot(X,Cubic_Polynomial,color='red')
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.show()
```

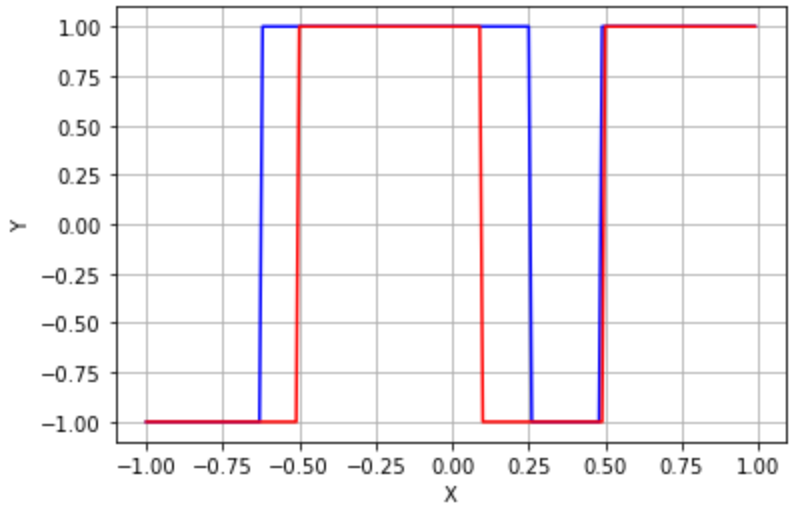


In [10]:
```
np.sign(Cubic_Polynomial)
```

Out[10]:
```
array([-1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1.,  1.,  1.])
```

Plotting the classifiers

In [11]:
```
plt.plot(X,np.sign(deg3pol),color="blue")
plt.plot(X,np.sign(Cubic_Polynomial),color='red')
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.show()
```



We can see that using the cubic polynomial and the given classifier we can perfectly classify the dataset.But we can not do that using degree 3 polynomial classifier.

In [ ]: