Solve for the ODE

$$\theta'' = \lambda\theta$$

where $\theta(0) = 1$ And $\theta'(1) = 0$ also $\lambda = 2$

In [25]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = [10, 15]
```

In [16]:

```python
def A(x):
    return 0
def B(x, lamda):
    return -lamda*x
def C(x):
    return 0

def get_a(x, h):
    return 1/(h**2) - A(x)/(2.0 * h)
def get_b(x, h, lamda):
    return - 2/(h**2) + B(x, lamda)
def get_c(x, h):
    return 1/(h**2) + A(x)/(2.0 * h)
```

In [17]:

```python
def ThomasAlgorithm(a, b, c, d, n):
    c_dash = np.zeros(n-1)
    d_dash = np.zeros(n-1)
    c_dash[0] = c[0] / b[0]
    d_dash[0] = d[0] / b[0]
    for itr in range(1, n-1):
        c_dash[itr] = c[itr] / (b[itr] - a[itr] * c_dash[itr-1])
        d_dash[itr] = (d[itr] - a[itr]*d_dash[itr-1]) / (b[itr] - a[itr] * c_dash[it

    y = np.zeros(n-1)
    y[n-2] = d_dash[n-2]

    for itr in reversed(range(n-2)):
        y[itr] = d_dash[itr] - c_dash[itr] * y[itr+1]

    return y
```

In [18]:

```python
def TridiagonalBVP(x0, xn, h, n, lamda):
    x = [(x0 + itr * h) for itr in range(1, n)]
    #print x[n-1]
    a = [get_a(itr, h) for itr in x]
    b = [get_b(itr, h, lamda) for itr in x]
    c = [get_c(itr, h) for itr in x]
    d = [C(itr) for itr in x]

    d[0] -= 1*a[0]
    b[n-2] += 4*c[n-2]/3
    a[n-2] += -1*c[n-2]/3

    return ThomasAlgorithm(a, b, c, d, n)

def func(x0, xn, h = 0.1):
    return np.arange(x0, xn + h, h)
```

In [24]:

```python
x0 = 0.0
xn = 1.5
steps = [0.25, 0.1, 0.02]
f = lambda xn, x0, step : int(np.ceil((xn-x0)/step))
colors = ['r', 'g', 'b']
lamda = 2
labels = ["step = 0.25", "step = 0.1", "step = 0.02"]
# Creating vector that stores the steps
for step in steps:
    y = []
    x_range = func(x0, xn, step)
#     print(x_range)
    n = int(np.ceil((xn - x0)/step))
#     print(n)
    y[1:n] = TridiagonalBVP(x0, xn, step, n, lamda)
#     print(y)
    y.insert(0, 1)
#     print(y)
    y.append((4 * y[-1] - y[-2])/3)
#     print(y)
    plt.xlabel('X')
    plt.ylabel('Y')
    i = steps.index(step)
    plt.plot( x_range, y, colors[i], label=labels[i])
    plt.gca().legend(labels)
```