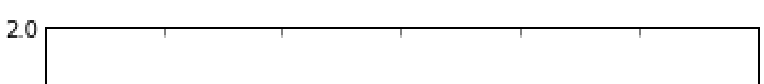
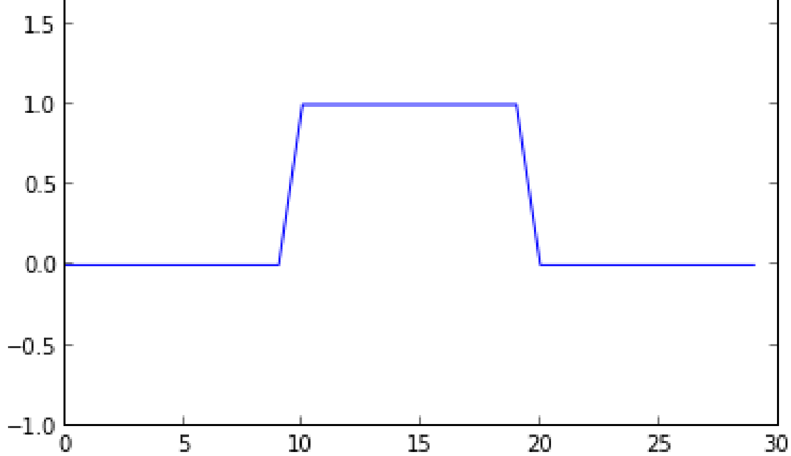
Reduce and Expand operations

```
Reduce - take a signal of length N and return a signal of length N/2.

Simple! Just take every other point.
Slightly less simple - smooth it first, then take every other point.

How to choose smoothing kernel w?

"The Generating Kernel"

- A special kernel that satisfies some nice constraints:

1) normalized (sums to 1)
2) symmetric
3) equal contribution - each pixel contributes 1/2 of the information to the next
   level.

For neighborhood size = 2,

w = [1/4-a/2, 1/4, a, 1/4, 1/4-a/2]

For a = 0.4, this closely approximates a Gaussian.
```

In [11]:
```
w = np.array([.05, .25, .4, .25, .05])
X = np.zeros(30)
X[10:20] = 1

plt.plot(X)
plt.ylim(-1, 2)
```

Out[11]: (-1, 2)

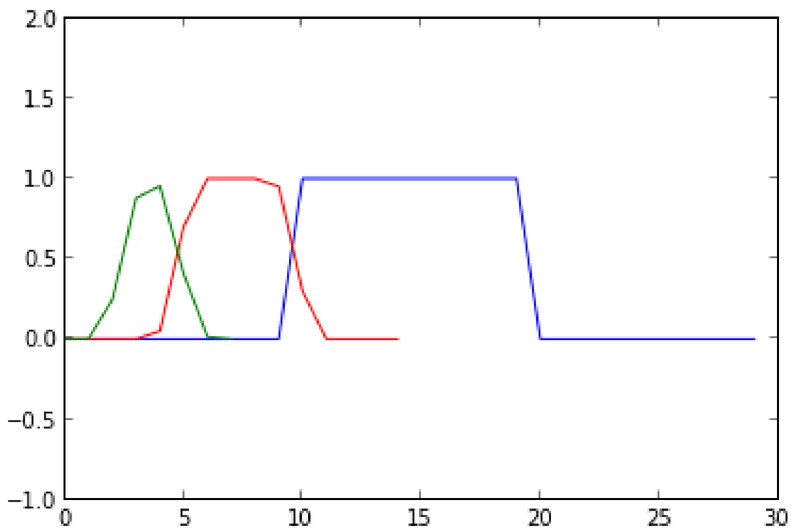




In [12]:

```
def reduce(X):
    out = np.convolve(X, w, 'same')
    return out[::2]

R_1 = reduce(X)
R_2 = reduce(R_1)

plt.plot(X, 'b', R_1, 'r', R_2, 'g')
plt.ylim(-1,2)
```

Out[12]: (-1, 2)



Expand - take a signal of length N and return a signal of length 2*N.

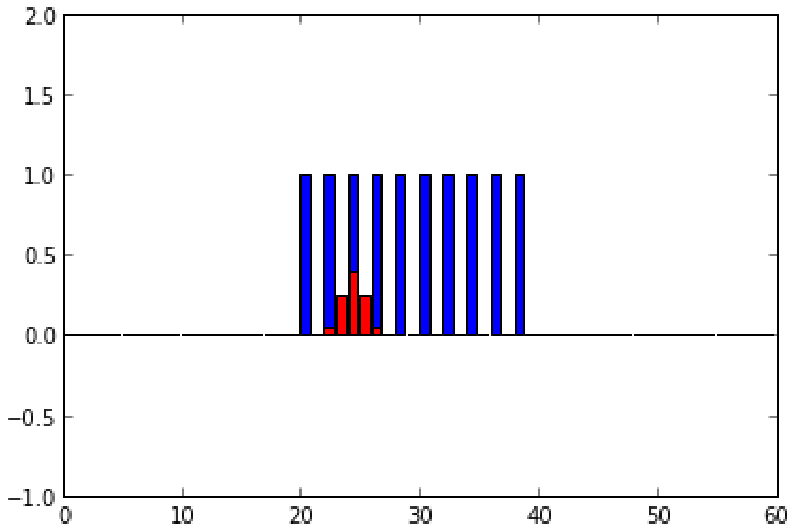
Uses the same kernel.
Opposite operation - sample, then smooth.

In [43]:

```
X_spaced = np.zeros(X.shape[0] * 2)
```

```
X_spaced[::2] = X[:]

bar(arange(X_spaced.shape[0]), X_spaced, color='blue')
bar(arange(22,27), w, color='red')
plt.xlim(0, 60)
plt.ylim(-1, 2)
```

Out[43]: (-1, 2)

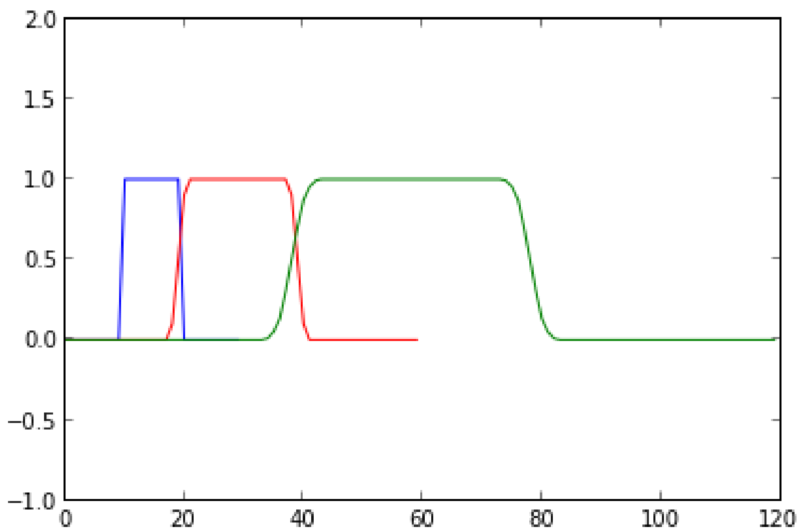



```
Think about the output at location 23.

It gets 0.4 of the third blue bar.
It gets 0.05 of the second blue bar.
It gets 0.05 of the fourth blue bar.

Total weight = 0.5

Consider the output at location 22.

It gets .25 of the second blue bar.
It gets .25 of the third blue bar.

Total weight = 0.5

(This is the equivalent contribution constraint from our kernel).

So, multiply the output by two!
```

In [45]:
```python
def expand(X):
    out = np.zeros(X.shape[0]*2)
    out[::2] = X
    return 2*np.convolve(out, w, 'same')
```

In [46]:
```python
E_1 = expand(X)
E_2 = expand(E_1)
plt.plot(X, 'b', E_1, 'r', E_2, 'g')
plt.ylim(-1,2)
```

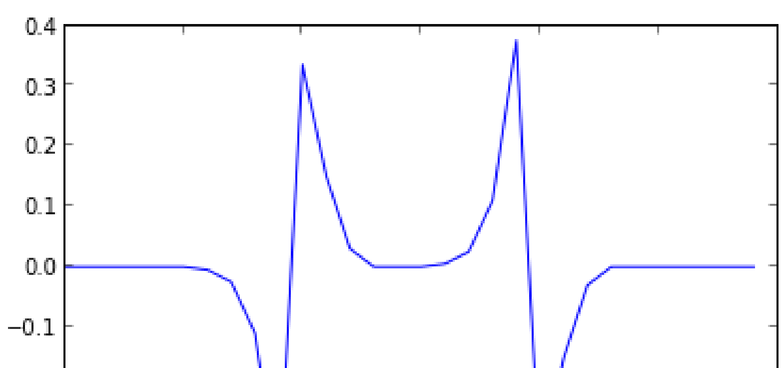
Out[46]: (-1, 2)

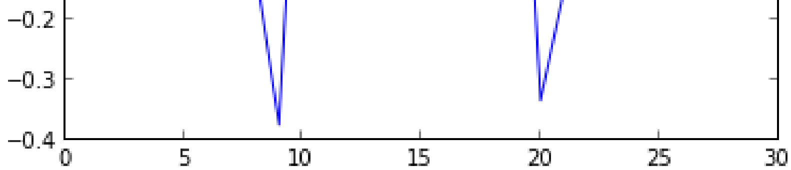


A quick note on laplacians:

In [49]:
```python
L = X - expand(reduce(X))
plt.plot(L)
```

Out[49]: [<matplotlib.lines.Line2D at 0x416c650>]






```
woot
```