Problem 2

In [ ]:

```
import numpy as np
import torch
from torch import nn
```

To compare with upsample module in torch, set both layers with kernel, stride = 2.

The mode 'nearest' in nn.Upsample send 1 to [[1,1],[1,1]], so we have to divide the result by 4. This is implemented in below.

As in problem 5 of homework 7, we identify 1×4 linear layer with 2×2 convolutional layer. Then our manual upsampling layer has weight 0.25 for all cell.

In [ ]:

```
upsample_layer_torch = nn.Upsample(scale_factor=2, mode='nearest')

def manual_upsampling(r : int):
    upsample_layer = nn.ConvTranspose2d(1, 1, kernel_size=r, stride=r, bias=False)
    upsample_layer.weight.data = torch.ones((r,r)).reshape(1,1,r,r)
    return upsample_layer
upsample_layer_manual = manual_upsampling(2)
```

In [ ]:

```
diff = torch.tensor(0)

with torch.no_grad():
    for _ in range(10):
        X = torch.rand(size=(3,3)).view(1,1,3,3)
        y_torch = upsample_layer_torch(X)
        y_manual = upsample_layer_manual(X)
        diff = torch.max(diff, torch.max(torch.abs(y_torch - y_manual)))

print(diff)
```

```
tensor(0.)
```

I run 10 randomly generated 3×3 target data. As you can see, outputs from both layers are exactly same.