4)

I worked on this code with Luca Leger and Alvin Delgado. How we did this was first we took an image and turned it into a 200X200 matrix. Then we used the built in SVD function of Python to create a function that would change the sigma matrix based on the input n. Next, we reconstructed the matrix by constructing an empty Sigma matrix full of zeros and populated the data from s (list of singular values) into its diagonal. Then we reconstructed the matrix using the new Sigma, which we used to generate the lower resolution images.

In [3]:
```
from matplotlib.image import imread
import matplotlib.pyplot as plt
import numpy as np
from numpy import diag
from numpy import dot
from numpy import zeros
from scipy.linalg import svd
import os

Image = imread('flower.jpg')
BandW = np.mean(Image, -1);

img = plt.imshow(BandW)
img.set_cmap('gray')
plt.axis('off')
plt.title('Original')
plt.show

BandW
```

Out[3]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 2., 0.],
       [0., 0., 0., ..., 4., 0., 1.],
       ...,
       [0., 0., 0., ..., 7., 7., 4.],
       [0., 0., 0., ..., 3., 0., 4.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [8]:
```
U, s, VT = svd(BandW)

def function1(sigma,n):
    for x in range (n,200,+1):
        sigma[x]=0
n = 150
function1(s,n)

Sigma = zeros((BandW.shape[0], BandW.shape[1]))
Sigma[:BandW.shape[1], :BandW.shape[1]] = diag(s)
BandW_compressed = U.dot(Sigma.dot(VT))

img = plt.imshow(BandW_compressed)
img.set_cmap('gray')
plt.axis('off')
plt.title('r = ' + str(n))
plt.show()

BandW_compressed
```

In [7]:
```
U, s, VT = svd(BandW)

def function1(sigma,n):
    for x in range (n,200,+1):
        sigma[x]=0
n = 100
function1(s,n)

Sigma = zeros((BandW.shape[0], BandW.shape[1]))
Sigma[:BandW.shape[1], :BandW.shape[1]] = diag(s)
BandW_compressed = U.dot(Sigma.dot(VT))

img = plt.imshow(BandW_compressed)
img.set_cmap('gray')
plt.axis('off')
plt.title('r = ' + str(n))
plt.show()

BandW_compressed
```

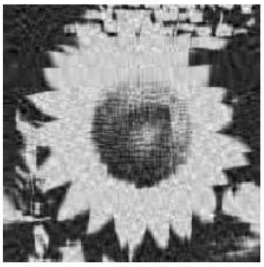
r = 100

In [6]:
```
U, s, VT = svd(BandW)

def function1(sigma,n):
    for x in range (n,200,+1):
        sigma[x]=0
n = 50
function1(s,n)

Sigma = zeros((BandW.shape[0], BandW.shape[1]))
Sigma[:BandW.shape[1], :BandW.shape[1]] = diag(s)
BandW_compressed = U.dot(Sigma.dot(VT))

img = plt.imshow(BandW_compressed)
img.set_cmap('gray')
plt.axis('off')
plt.title('r = ' + str(n))
plt.show()

BandW_compressed
```

r = 50

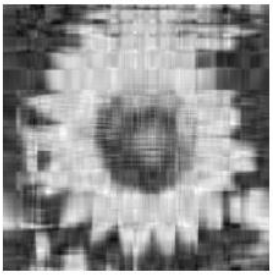
In [5]:
```
U, s, VT = svd(BandW)

def function1(sigma,n):
    for x in range (n,200,+1):
        sigma[x]=0
n = 25
function1(s,n)

Sigma = zeros((BandW.shape[0], BandW.shape[1]))
Sigma[:BandW.shape[1], :BandW.shape[1]] = diag(s)
BandW_compressed = U.dot(Sigma.dot(VT))

img = plt.imshow(BandW_compressed)
img.set_cmap('gray')
plt.axis('off')
plt.title('r = ' + str(n))
plt.show()

BandW_compressed
```

r = 25



In [4]:
```
U, s, VT = svd(BandW)

def function1(sigma,n):
    for x in range (n,200,+1):
        sigma[x]=0
n = 10
function1(s,n)

Sigma = zeros((BandW.shape[0], BandW.shape[1]))
Sigma[:BandW.shape[1], :BandW.shape[1]] = diag(s)
BandW_compressed = U.dot(Sigma.dot(VT))

img = plt.imshow(BandW_compressed)
img.set_cmap('gray')
plt.axis('off')
plt.title('r = ' + str(n))
plt.show()

BandW_compressed
```

r = 10



We consulted the following website for tutorial on how to reconstruct SVD matrices.

Sources: https://machinelearningmastery.com/singular-value-decomposition-for-machine-learning/