1. (a).

The minimum value of objective is 0.
Because when $k=n$, one cluster has one point and every point is in its own cluster.

(b) Let $f(\mu_i) = \left(\sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 \right) + \lambda \|\mu_i\|_2^2,$

Take gradient of $f(\mu_j)$ with respect to μ_i

$$\nabla_{\mu_i} f(\mu_i) = \left(2 \sum_{x_j \in C_i} (\mu_i - x_j)\right) + 2\lambda \mu_i$$

$$\nabla_{\mu_i} f(\mu_i) = 2\left((|C_i| + \lambda)\mu_i - \sum_{x_j \in C_i} x_j\right)$$

Set $\nabla_{\mu_i} f(\mu_i) = 0$, we get

$$\mu_i = \frac{1}{|C_i| + \lambda} \sum_{x_j \in C_i} x_j$$

Scanned by CamScanner

(c)

$$\min_{\mu_i \in \mathbb{R}^d} \sum_{i=1}^{k} \left(\|\mu_i\|_2 + \sum_{x_j \in C_i} \|x_j - \mu_j\|_2 \right)$$

(d) Given cluster S_i,

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} \phi(x)$$

to get optimal clustering, we have to minimize:

$$\sum_{x \in S_i} \|\phi(x_i) - \mu_k\|^2$$

Let $f(i,k) = \|\phi(x_i) - \mu_k\|^2$

$$= \langle \phi(x_i), \phi(x_i) \rangle - 2 \langle \phi(x_i), \mu_k \rangle + \langle \mu_k, \mu_k \rangle$$

Plug in $\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} \phi(x)$, we get

$$f(i,k) = \langle \phi(x_i), \phi(x_i) \rangle - \frac{2}{|S_i|} \sum_{x_j \in S_k} \langle \phi(x_i), \phi(x_j) \rangle + \frac{1}{|S_k|^2} \sum_{x_j, x_l \in S_k \times S_k} \langle \phi(x_j), \phi(x_l) \rangle$$

$$= K(x_i, x_i) - \frac{2}{|S_i|} \sum_{x_j \in S_k} K(x_i, x_j) + \frac{1}{|S_k|^2} \sum_{x_j, x_l \in S_k \times S_k} K(x_j, x_l)$$

So, $\text{class}(i) = \arg\min_k \; -\frac{2}{|S_i|} \sum_{x_j \in S_k} K(x_i, x_j) + \frac{1}{|S_k|^2} \sum_{x_j, x_l \in S_k \times S_k} K(x_j, x_l)$

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from imageio import imread

%matplotlib inline
```

Part a)

In [32]:

```
def low_rank_approximation(X, rank):
    # YOUR CODE GOES HERE
    u, s, vt = np.linalg.svd(X)
    s[rank:] = 0
    s = np.vstack((np.diag(s), np.zeros((u.shape[1]-s.shape[0], vt.shape[0]))))
    return u@s@vt
```

In [31]:

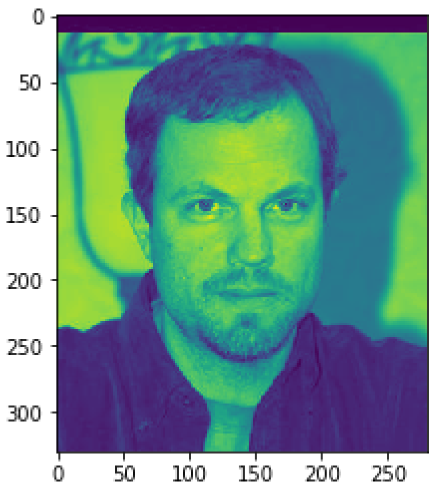
```
u, s, vt = np.linalg.svd(face)
s[5:] = 0
s = np.vstack((np.diag(s), np.zeros((u.shape[1]-s.shape[0], vt.shape[0]))))
result = u@s@vt
u.shape,s.shape,vt.shape
```

Out[31]:

```
((330, 330), (330, 280), (280, 280))
```

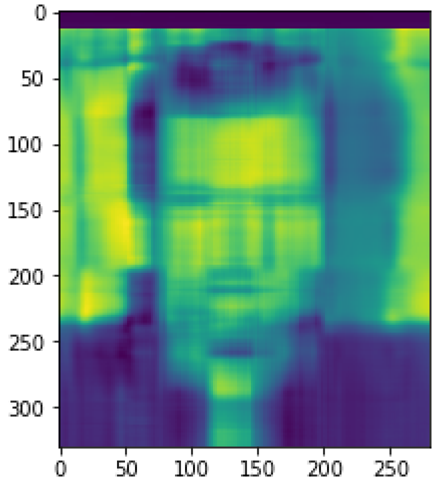
In [36]:

```
# original picture
face = imread("./data/face.jpg")
plt.imshow(face)
plt.show()
```



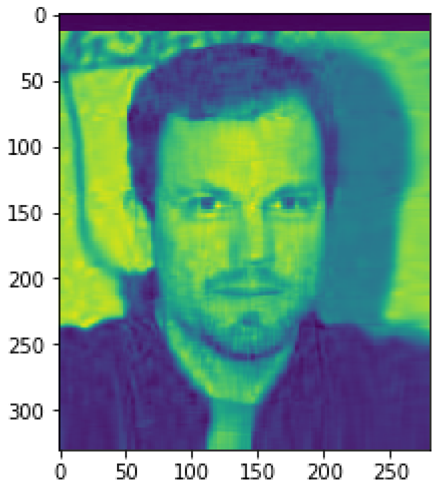
In [33]:

```
# rank = 5
face_rank5 = low_rank_approximation(face, 5)
plt.imshow(face_rank5)
plt.show()
```



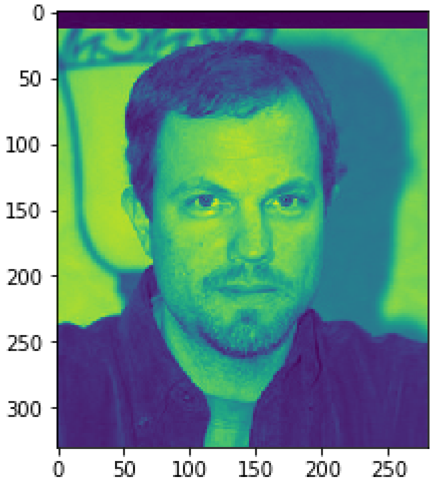
In [34]:

```
# rank = 20
face_rank20 = low_rank_approximation(face, 20)
plt.imshow(face_rank20)
plt.show()
```



In [35]:

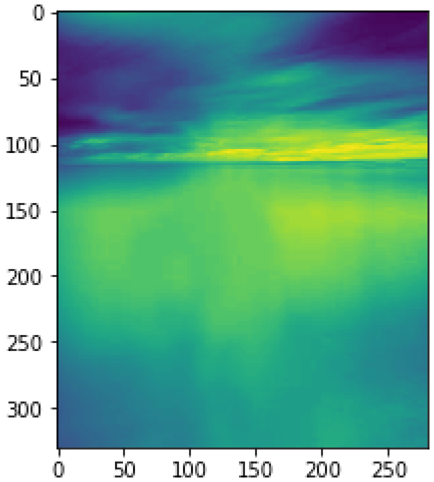
```
# rank = 100
face_rank100 = low_rank_approximation(face, 100)
plt.imshow(face_rank100)
plt.show()
```



Part b)

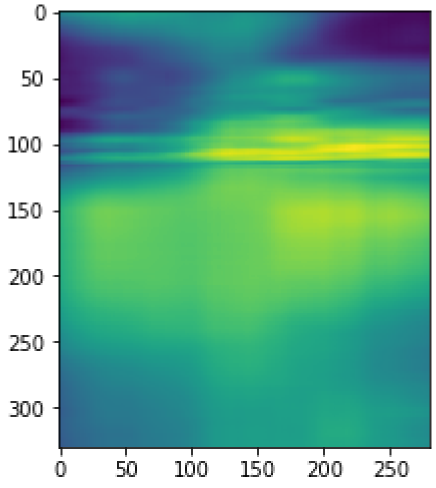
In [38]:

```
# original picture
sky = imread("./data/sky.jpg")
plt.imshow(sky)
plt.show()
```



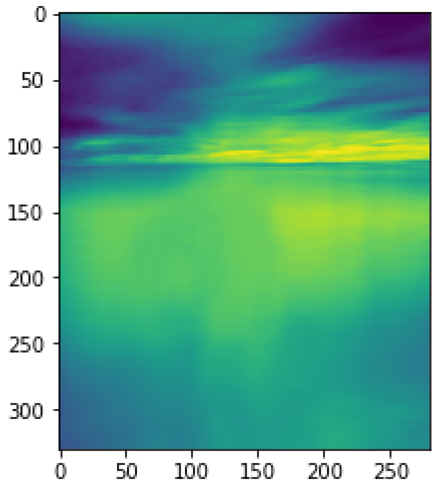
In [42]:

```
# rank = 5
sky_rank5 = low_rank_approximation(sky, 5)
plt.imshow(sky_rank5)
plt.show()
```



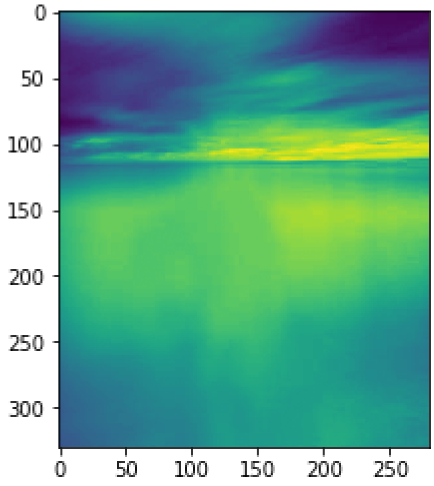
In [43]:

```
# rank = 20
sky_rank20 = low_rank_approximation(sky, 20)
plt.imshow(sky_rank20)
plt.show()
```



In [44]:

```
# rank = 100
sky_rank100 = low_rank_approximation(sky, 100)
plt.imshow(sky_rank100)
plt.show()
```

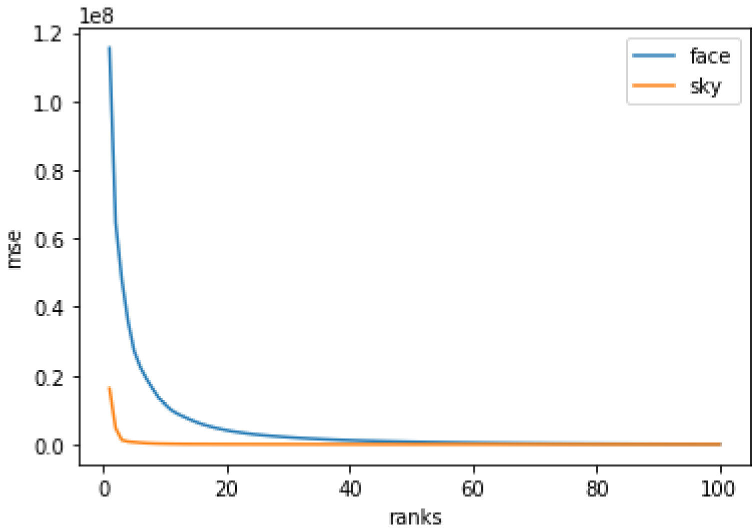


Part c)

In [53]:

```
def mse(img1, img2):
    # YOUR CODE GOES HERE
    face_MSE = []
    sky_MSE = []
    for i, r in enumerate(range(1,101)):
        face_rank = low_rank_approximation(img1, r)
        sky_rank = low_rank_approximation(img2, r)
        face_MSE.append(np.sum(np.square(face_rank-img1)))
        sky_MSE.append(np.sum(np.square(sky_rank-img2)))
    plt.plot(range(1,101), face_MSE,label = 'face')
    plt.plot(range(1,101), sky_MSE,label = 'sky')
    plt.xlabel('ranks')
    plt.ylabel('mse')
    plt.legend()
    plt.show()

mse(face, sky)
plt.savefig('mse.png')
```

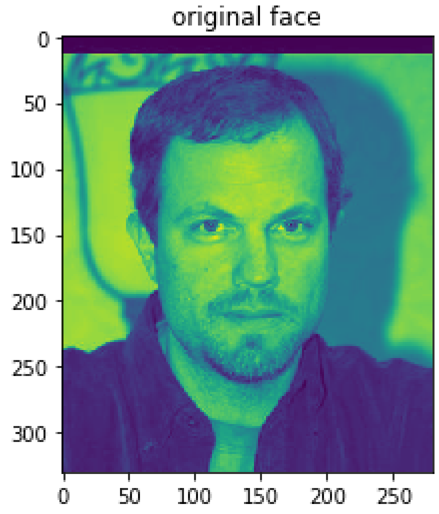


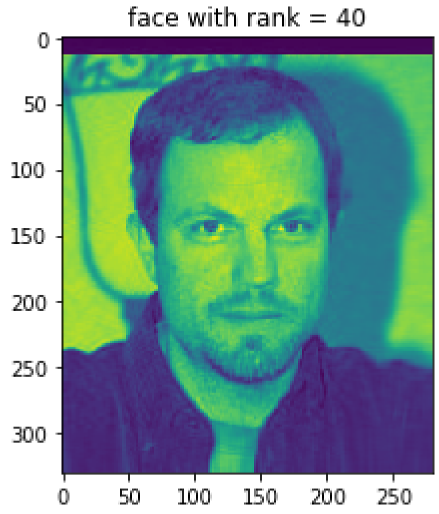

```
<Figure size 432x288 with 0 Axes>
```

In [64]:

```
plt.imshow(face)
plt.title('original face')
plt.show()

face_rank100 = low_rank_approximation(face, 40)
plt.imshow(face_rank100)
plt.title('face with rank = 40')
plt.show()
```

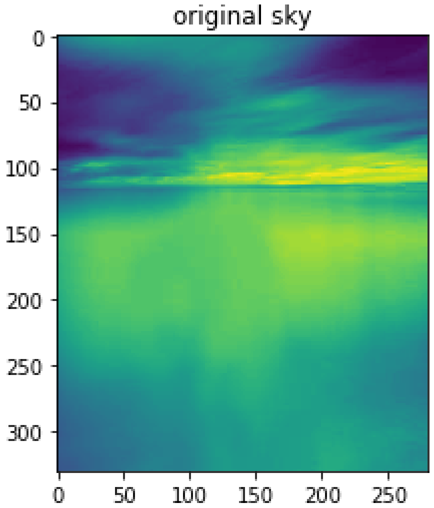


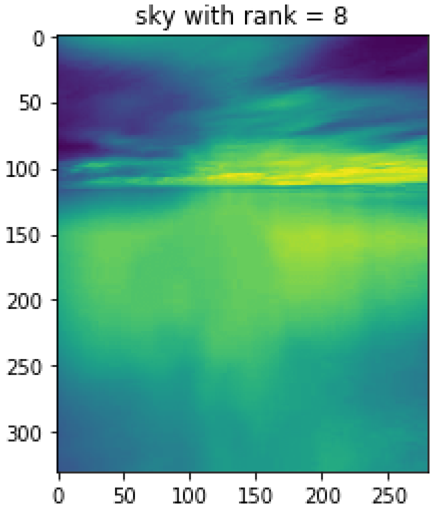


In [63]:

```
plt.imshow(sky)
plt.title('original sky')
plt.show()

face_rank100 = low_rank_approximation(sky, 8)
plt.imshow(sky_rank100)
plt.title('sky with rank = 8')
plt.show()
```





(d) The lowest rank approximation of face is 40, the lowest rank approximation of sky is 8. The reason is that face picture has more details than sky picture, which need more components to approximate.

I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted.

Di Zhen