```python
import numpy as np
import cv2
import os


#A class that stores all the images including input image, smoothed image,
#sharped image and final improved image. The class also has functions that
#apply different filters for smoothing and sharpening according to choice
class Solution():
    #constructor that declares and assigns the input image
    def __init__(self,filename):
        self.input_image = cv2.imread(filename)

    #This function smoothes the image according to choice and assigns it to
    #the member variable self.smoothed_image
    def smooth(self,smooth_type,image):
        #uses gaussian filter of kernel 3
        if (smooth_type == 'GaussianBlur'):
            self.smoothed_image = cv2.GaussianBlur(image,(3,3),sigmaX=2)

        #applies median filter of size 3
        elif (smooth_type == 'medianBlur'):
            self.smoothed_image = cv2.medianBlur(image,3)

        #applies boxFilter of size 3, -1 indicates input and output of same dimensions
        elif (smooth_type == 'boxFilter'):
            self.smoothed_image = cv2.boxFilter(image,-1,(3,3))

        #applies the normal blur which takes average around the kernel of size(3,3)
        elif (smooth_type == 'blur'):
            self.smoothed_image = cv2.blur(image,(3,3))

        #applies the bilateral filter of diameter 15, sigma of color space & coordinate
        #space respectively
        elif (smooth_type == 'bilateralFilter'):
            self.smoothed_image = cv2.bilateralFilter(image,15,75,75)

        elif (smooth_type == 'unsharpMasking'):
            temp_img = cv2.GaussianBlur(image,(3,3),sigmaX=1)
            self.smoothed_image = cv2.addWeighted(temp_img,-.5,image,1.7,0)

        else :
            raise Exception("Invalid choice")
```

```
    #This function sharps the image according to the sharpening matrix of choice
    #and assigns the sharped image to self.sharp_image
    def sharp(self,sharp_matrix,image):

        #sharps by taking the sharp_matrix and image as input
        self.sharp_image = cv2.filter2D(image,-1,sharp_matrix)


def main():

    #For image 1, pcb.png
    filename = "/home/akshay/Downloads/CV/ps-3-q/ps3-images/pcb.png"
    #Creating an object for the pcb image
    image_improve_pcb = Solution(filename)
    #smoothing it using median filter
    image_improve_pcb.smooth('medianBlur',image_improve_pcb.input_image)
    #sharpening the smoothed it using the folowing matrix
    sharp_matrix = np.asarray([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
    image_improve_pcb.sharp(sharp_matrix,image_improve_pcb.smoothed_image)

    #writing the image
    first_name = os.path.basename(filename)
    first_name = first_name.split('.',1)[0]
    write_filename = first_name + "-improved.png"

    if(cv2.imwrite(write_filename,image_improve_pcb.sharp_image)):
        print("Successfully saved:",write_filename)
    else:
        print("Save Unsuccessfull")
    #printing the input as well as improved image
    cv2.imshow("input image pcb",image_improve_pcb.input_image)
    cv2.imshow("final improved image pcb",image_improve_pcb.sharp_image)


    #for image 2 golf.png
    filename = "/home/akshay/Downloads/CV/ps-3-q/ps3-images/golf.png"
    #Creating an object for the pcb image
    image_improve_golf = Solution(filename)
    #smoothing it using median filter
    image_improve_golf.smooth('medianBlur',image_improve_golf.input_image)
    #sharpening the smoothed it using the folowing matrix
    sharp_matrix = np.asarray([[0,-1,0],[-1,5,-1],[0,-1,0]])
    image_improve_golf.sharp(sharp_matrix,image_improve_golf.smoothed_image)

    #writing the image
```

```
first_name = os.path.basename(filename)
first_name = first_name.split('.',1)[0]
write_filename = first_name + "-improved.png"

if(cv2.imwrite(write_filename,image_improve_golf.sharp_image)):
    print("Successfully saved: ",write_filename)
else:
    print("Save Unsuccessfull")

#printing the input as well as improved image
cv2.imshow("input image golf",image_improve_golf.input_image)
cv2.imshow("final improved image golf",image_improve_golf.sharp_image)


#for image 3 pots.png
filename = "/home/akshay/Downloads/CV/ps-3-q/ps3-images/pots.png"
#Creating an object for the pots image
image_improve_pots = Solution(filename)
#smoothing it using median filter
#image_improve_pots.smooth('unsharpMasking',image_improve_pots.input_image)
#sharpening the smoothed it using the folowing matrix
sharp_matrix = np.asarray([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
image_improve_pots.sharp(sharp_matrix,image_improve_pots.input_image)
image_improve_pots.smooth('unsharpMasking',image_improve_pots.sharp_image)

#writing the image
first_name = os.path.basename(filename)
first_name = first_name.split('.',1)[0]
write_filename = first_name + "-improved.png"

if(cv2.imwrite(write_filename,image_improve_pots.smoothed_image)):
    print("Successfully saved: ",write_filename)
else:
    print("Save Unsuccessfull")

#printing the input as well as improved image
cv2.imshow("input image pots",image_improve_pots.input_image)
cv2.imshow("final improved image pots",image_improve_pots.smoothed_image)


#for image 4 rainbow.png
filename = "/home/akshay/Downloads/CV/ps-3-q/ps3-images/rainbow.png"
#Creating an object for the rainbow image
image_improve_rainbow = Solution(filename)
#smoothing it using median filter
```

```
    image_improve_rainbow.smooth('bilateralFilter',image_improve_rainbow.input_image)
    #sharpening the smoothed it using the folowing matrix
    sharp_matrix = np.asarray([[0,-1,0],[-1,5,-1],[0,-1,0]])
    image_improve_rainbow.sharp(sharp_matrix,image_improve_rainbow.smoothed_image)

    #writing the image
    first_name = os.path.basename(filename)
    first_name = first_name.split('.',1)[0]
    write_filename = first_name + "-improved.png"

    if(cv2.imwrite(write_filename,image_improve_rainbow.sharp_image)):
        print("Successfully saved: ",write_filename)
    else:
        print("Save Unsuccessfull")


    #printing the input as well as improved image
    cv2.imshow("input image rainbow",image_improve_rainbow.input_image)
    cv2.imshow("final improved image rainbow",image_improve_rainbow.sharp_image)


if __name__ == '__main__':
    main()
    cv2.waitKey(0)
```