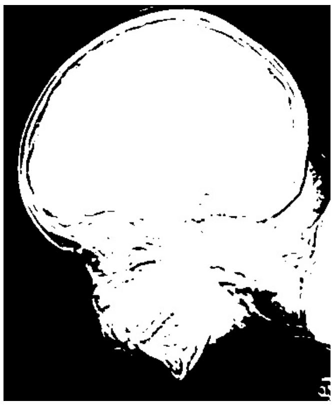
DIP homework 1

4109061012 陳柏翔

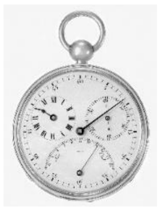
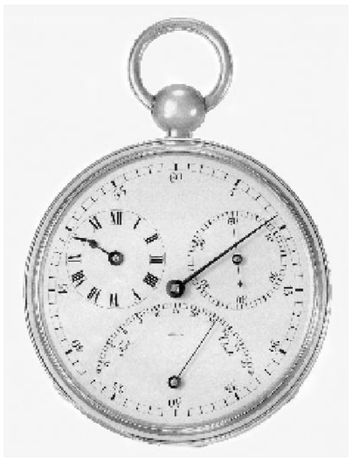
Problem 8：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np

FILE_PATH = "homework_1/src/Fig0221(a)(ctskull-256).tif"
NEW_LEVELS = 2
I_MAX = 2**8 - 1  # original max intensity value

def reduce_intensity_levels(img: np.ndarray, levels: int) -> np.ndarray:
    """ Reduce the number of intensity levels in an image. """
    if levels <= 0: raise ValueError("the new number of intensity levels must > 0")
    elif levels == 1: return np.zeros_like(img)

    interval = I_MAX / levels
    intensities = (np.arange(levels) * I_MAX / (levels - 1)).astype(int)
    print(f">> new intensities ({levels} levels): \n{intensities}\n")

    for i in range(levels):
        img[(img >= interval * i) & (img < interval * (i+1))] = intensities[i]
    return img

if __name__ == "__main__":
    img = np.asarray(cv2.imread(FILE_PATH, cv2.IMREAD_GRAYSCALE))
    img = reduce_intensity_levels(img, NEW_LEVELS)
    cv2.imwrite("problem_8(b)_output.jpg", img)

    print(">> done.")
```


(Result)

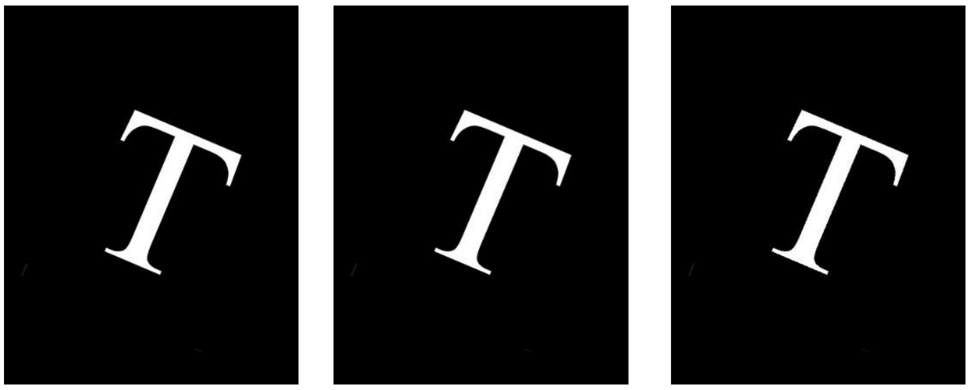
Problem 9：

```
# 4109061012 B.S.Chen
import cv2

FILE_PATH = "homework_1/src/Fig0220(a)(chronometer 3692x2812  2pt25 inch 1250 dpi).tif"
FACTOR = 12

if __name__ == "__main__":
    """ Shrink the image by a factor first, then zoom back to original size. """
    img = cv2.imread(FILE_PATH)
    orig_size = (img.shape[1], img.shape[0])
    shrink_size = (round(img.shape[1] / FACTOR), round(img.shape[0] / FACTOR))

    # Shrinking
    img = cv2.resize(img, shrink_size, interpolation=cv2.INTER_NEAREST)
    cv2.imwrite("problem_9(b)_output.jpg", img)

    # Zoom back to original size
    img = cv2.resize(img, orig_size, interpolation=cv2.INTER_NEAREST)
    cv2.imwrite("problem_9(c)_output.jpg", img)

    print(">> done")
```



(Results) 非真實比例

Problem 10：

```python
# 4109061012 B.S.Chen
import cv2


FILE_PATH = "homework_1/src/Fig0236(a)(letter_T).tif"
ROTATE = -23  # ccw
SCALE = 2 / 3
SHIFT = (18, 22)  # x, y


def image_RSTI(img: cv2.Mat, rotate = 0.0, scale = 1.0, translate = (0, 0), interpolation = "nearest"):
    """ Apply rotating, scaling, translating(shifting), and using different interpolation on an image. """
    if interpolation not in ("nearest", "bilinear", "bicubic"):
        raise ValueError(f"interpolation must be \"nearest\", \"bilinear\", \"bicubic\"")

    center = (img.shape[1] // 2 + translate[0], img.shape[0] // 2 + translate[1])
    transform = cv2.getRotationMatrix2D(center, rotate, scale)
    if interpolation == "nearest":
        return cv2.warpAffine(img, transform, (img.shape[1], img.shape[0]), flags=cv2.INTER_NEAREST)
    elif interpolation == "bilinear":
        return cv2.warpAffine(img, transform, (img.shape[1], img.shape[0]), flags=cv2.INTER_LINEAR)
    else:
        return cv2.warpAffine(img, transform, (img.shape[1], img.shape[0]), flags=cv2.INTER_CUBIC)


if __name__ == "__main__":
    img = cv2.imread(FILE_PATH)
    nearest_img = image_RSTI(img, rotate=ROTATE, scale=SCALE, translate=SHIFT, interpolation="nearest")
    bilinear_img = image_RSTI(img, rotate=ROTATE, scale=SCALE, translate=SHIFT, interpolation="bilinear")
    bicubic_img = image_RSTI(img, rotate=ROTATE, scale=SCALE, translate=SHIFT, interpolation="bicubic")

    cv2.imwrite("problem_10(b)_nearest.jpg", nearest_img)
    cv2.imwrite("problem_10(b)_bilinear.jpg", bilinear_img)
    cv2.imwrite("problem_10(b)_bicubic.jpg", bicubic_img)

    print(">> done")
```



(Results) bicubic / bilinear / nearest