**Прозрачность**

Два изображения можно совместить так, что одно из них будет как бы «полупрозрачным». Для этого значения цветовых компонент каждого совмещаемого пикселя обоих изображений нужно попарно суммировать с определёнными весовыми коэффициентами.  
  
Например, если итоговый цвет (в нотации RGB) вычислять по формуле:

   R = 0.8 \* R1 + 0.2 \* R2    
   G = 0.8 \* G1 + 0.2 \* G2    
   B = 0.8 \* B1 + 0.2 \* B2

то получится, что первое изображение будет иметь 20 процентов прозрачности.  
  
Продемонстрируем эффект прозрачности:  
  
  
  
  
  
  
  
Напишите функцию **transparency(filename1, filename2)**, которая, используя PIL, смешивает изображения в пропорции 50/50.

**Формат ввода**

Тестирующая система вызовет вашу функцию, передав в неё имена файлов с изображениями:

   transparency("image1.jpg", "image2.jpg")

**Формат вывода**

Результат смешивания изображений сохраните в файл с именем **res.jpg**в той же папке, где находится ваша программа.

**Примечания**

Результат вычисления каждой компоненты нового пикселя необходимо приводить к **целому**числу с помощью функции **int**.  
Например, так:

   r = int(0.2 \* r1 + 0.8 \* r2)

**Решение.**

from PIL import Image

def transparency(filename1, filename2):

im1 = Image.open(filename1)

im2 = Image.open(filename2)

x, y = im1.size

pixels1 = im1.load()

pixels2 = im2.load()

for i in range(x):

for j in range(y):

r1, g1, b1 = pixels1[i, j]

r2, g2, b2 = pixels2[i, j]

r = int(0.5 \* r1 + 0.5 \* r2)

g = int(0.5 \* g1 + 0.5 \* g2)

b = int(0.5 \* b1 + 0.5 \* b2)

pixels1[i, j] = r, g, b

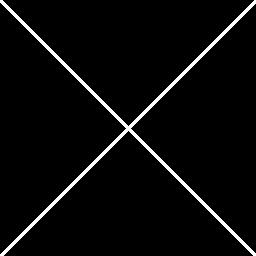
im1.save('res.jpg')

**Графический миксер**

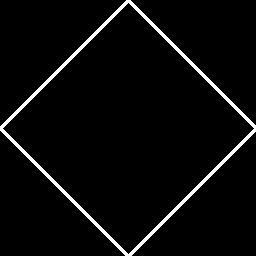
Напишите функцию twist\_image(input\_file\_name, output\_file\_name), которая будет менять местами левую и правую половины изображения.

Параметр input\_file\_name задаёт имя исходного файла, а output\_file\_name — имя файла, куда следует сохранить результат.

При передаче данного изображения:



должно получиться:



**Формат ввода**

Функция не должна ничего считывать.

**Формат вывода**

Функция не должна ничего выводить.

**Примечания**

Все имена файлов имеют расширение .jpg.

from PIL import Image

def twist\_image(input\_file\_name, output\_file\_name):

im = Image.open(input\_file\_name)

x, y = im.size

im2 = im.crop((0, 0, x // 2, y))

im3 = im.crop((x // 2, 0, x, y))

im.paste(im3)

im.paste(im2, (x // 2, 0, x, y))

im.save(output\_file\_name)

# Поворот с размытием

В файле **image.jpg** лежит изображение. Напишите функцию **motion\_blur(n)** которая:

1. открывает изображение **image.jpg**
2. поворачивает его на 270 градусов против часовой стрелки без использования циклов (только встроенными средствами PIL)
3. обрабатывает полученное изображение с помощью размытия Гаусса (**GaussianBlur**) с параметром **n**
4. сохраняет результат в файле **res.jpg**

## Формат ввода

Изображение **image.jpg** в текущей папке



вызов функции:  
**motion\_blur(10)**

## Формат вывода

Изображение **res.jpg** в текущей папке



from PIL import Image, ImageFilter

def motion\_blur(n):

image = Image.open('image.jpg')

image = image.transpose(Image.ROTATE\_270)

image = image.filter(ImageFilter.GaussianBlur(radius=n))

image.save('res.jpg')

**Миниатюра для сайта**

В файле **image.jpg**лежит изображение. Напишите функцию **make\_preview(size,** **n\_colors)**которая:

1. открывает изображение **image.jpg**
2. уменьшает его до размера **size**
3. уменьшает число цветов в изображении до **n\_colors**(такая процедура называется квантование или **quantize**)
4. сохраняет результат в файле **res.bmp**

**Формат ввода**

Изображение **image.jpg**в текущей папке.



вызов функции:

make\_preview((400, 200), 64)

**Формат вывода**



**Примечания**

Обратите внимание, что результат сохраняется в формате **bmp**.

from PIL import Image

def make\_preview(size, n\_colors: int):

image = Image.open('image.jpg')

image = image.resize(size)

image = image.quantize(n\_colors)

image.save('res.bmp')