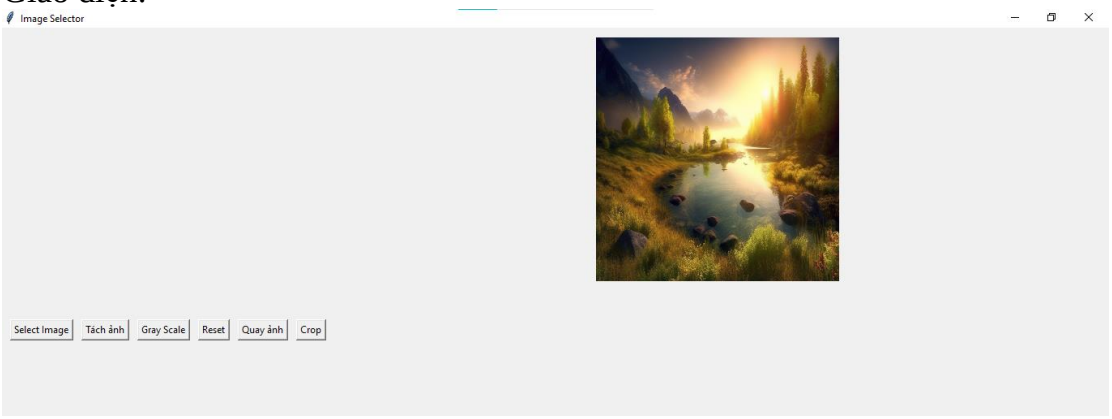
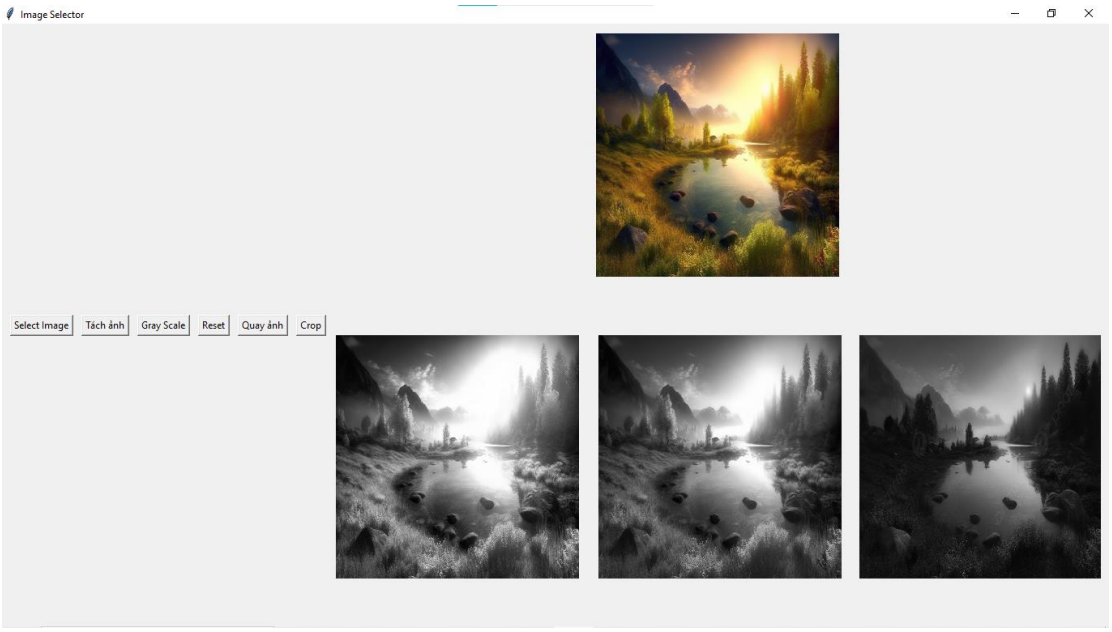
21110185_HoVanHuynhHop_BtTuan01

HW1) Thu thập (tìm trên mạng hoặc tự chụp ảnh) tối thiểu 10 ảnh phong cảnh (độ phân giải 566x1080 pixels), sau đó lưu mỗi ảnh thành 3 file theo các định dạng PNG, BMP, JPG.
HW2) Viết chương trình bằng Python hoặc C++
• Đọc ảnh và hiển thị các ảnh đã tạo ở HW1.
• Tách ảnh RGB thành 3 lớp R, G, B và hiển thị chúng
• Chuyển ảnh RGB thành ảnh đa mức xám (gray scale)
• Read image set and show each image on each window.
• Quay ảnh 100 lần: mỗi lần quay 10 độ, và tạm dừng 0.1 giây, hiển thị trên cùng cửa sổ.
• Đọc vào 1 ảnh, sau đó crop 1⁄4 ảnh tính từ tâm ảnh và hiển thị ảnh trước và sau khi crop.

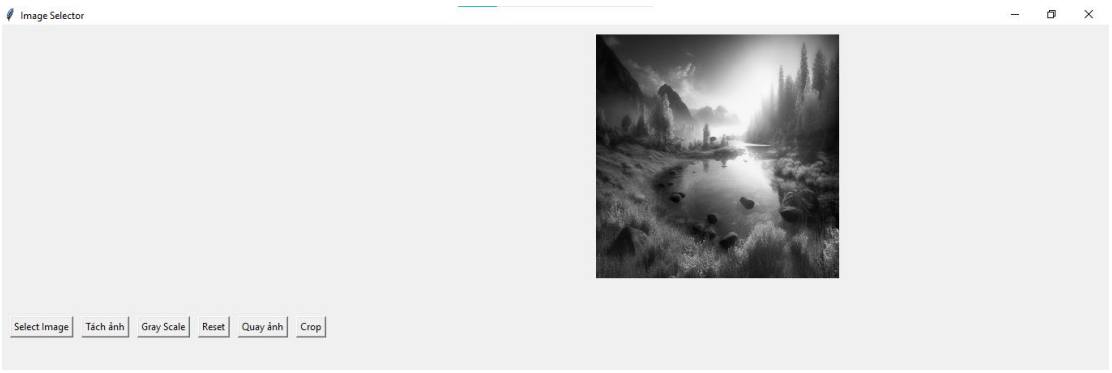
*Kết quả:
Giao diện:



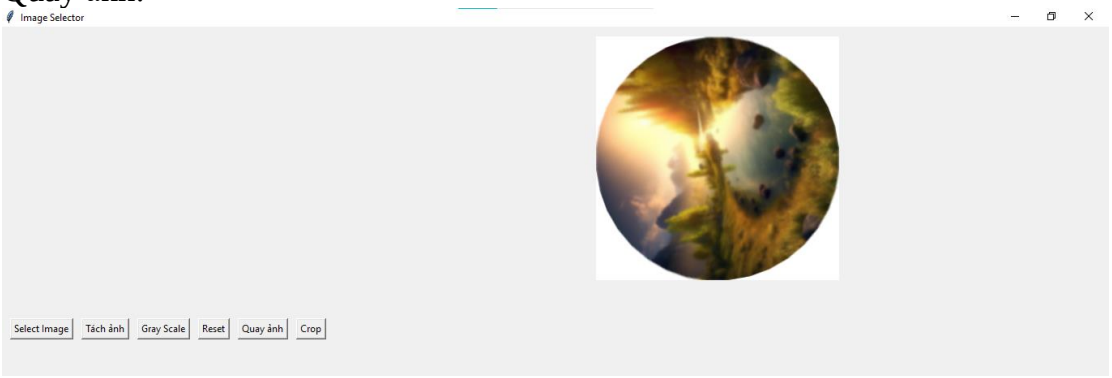
Tách ảnh:



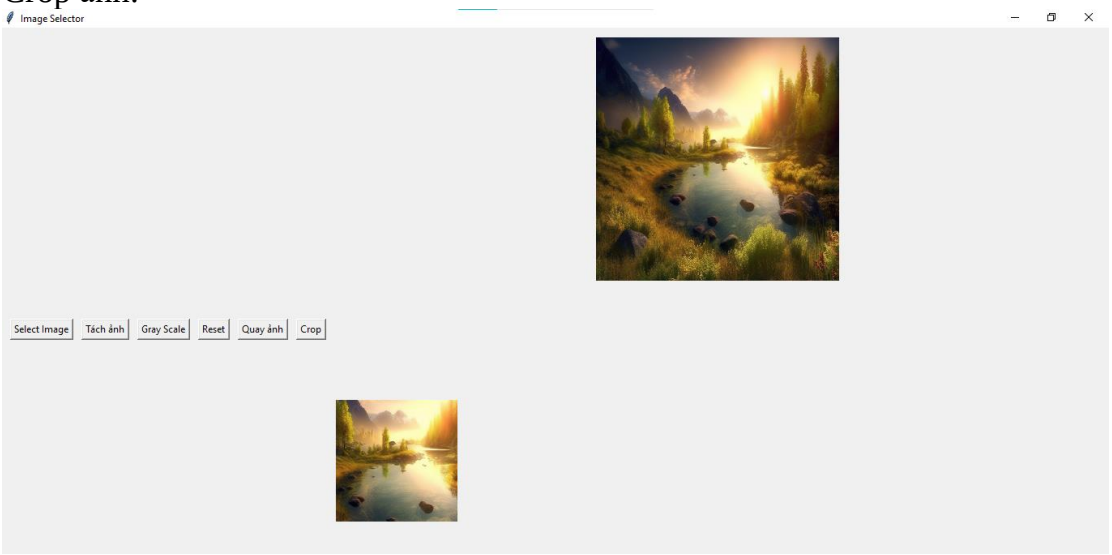
Gray Scale:



Quay ảnh:



Crop ảnh:



* Code:

```
import cv2
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import filedialog
import time
import numpy as np
from threading import Thread

class ImageSelectorApp:
    def __init__(self, master):
```

```
        self.master = master
        self.master.title("Image Selector")

        # Tạo nút "Select Image" để chọn ảnh
        self.select_button = tk.Button(master, text="Select Image",
command=self.choose_image)
        self.select_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Tách ảnh" để thực hiện quá trình tách ảnh
        self.split_button = tk.Button(master, text="Tách ảnh",
command=self.split_image)
        self.split_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Gray Scale" để chuyển đổi ảnh thành ảnh đa mức xám
        self.gray_button = tk.Button(master, text="Gray Scale",
command=self.convert_to_gray)
        self.gray_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Reset" để xóa toàn bộ ảnh đã hiển thị
        self.reset_button = tk.Button(master, text="Reset",
command=self.reset_images)
        self.reset_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Quay ảnh" để thực hiện chức năng quay ảnh
        self.rotate_button = tk.Button(master, text="Quay ảnh",
command=self.rotate_images)
        self.rotate_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo nút "Crop" để cắt ảnh
        self.crop_button = tk.Button(master, text="Crop",
command=self.crop_image)
        self.crop_button.pack(side="left", pady=10, padx=(10, 0))

        # Tạo label để hiển thị ảnh gốc
        self.original_image_label = tk.Label(master)
        self.original_image_label.pack(pady=10, padx=10)  # Điều chỉnh giảm
đệm

        # Tạo label để hiển thị các lớp R, G, B
        self.r_label = tk.Label(master)
        self.r_label.pack(side="left", pady=10, padx=10)

        self.g_label = tk.Label(master)
        self.g_label.pack(side="left", pady=10, padx=10)

        self.b_label = tk.Label(master)
```

```
        self.b_label.pack(side="left", pady=10, padx=10)

        # Biến instance để kiểm tra trạng thái quay ảnh
        self.is_rotating = False

        # Biến instance để lưu ảnh gốc
        self.original_image = None

        # Tiến trình quay ảnh
        self.rotation_process = None

    # Hàm để chọn ảnh từ thư mục
    def choose_image(self):
        file_path = filedialog.askopenfilename()
        if file_path:
            self.display_original_image(file_path)

    # Hàm để hiển thị ảnh gốc
    def display_original_image(self, file_path):
        # Đọc ảnh sử dụng OpenCV
        original_image = cv2.imread(file_path)
        original_image = cv2.cvtColor(original_image,
cv2.COLOR_BGR2RGB)  # Chuyển đổi sang màu RGB

        # Thiết lập kích thước ảnh cố định (ví dụ: 300x300)
        target_size = (300, 300)
        original_image = cv2.resize(original_image, target_size)

        # Lưu ảnh gốc vào biến instance
        self.original_image = original_image

        # Hiển thị ảnh gốc trong label
        original_image_pil = Image.fromarray(original_image)
        original_image_tk = ImageTk.PhotoImage(original_image_pil)
        self.original_image_label.config(image=original_image_tk)
        self.original_image_label.image = original_image_tk

    # Hàm để tách ảnh thành các lớp R, G, B và hiển thị
    def split_image(self):
        # Kiểm tra xem đã chọn ảnh chưa
        if self.original_image is None:
            return

        # Lấy ảnh gốc từ biến instance
        original_image = self.original_image
```

```
        # Tách thành các lớp R, G, B
        r, g, b = cv2.split(original_image)

        # Hiển thị các lớp R, G, B trong label
        r_pil = Image.fromarray(r)
        r_tk = ImageTk.PhotoImage(r_pil)
        self.r_label.config(image=r_tk)
        self.r_label.image = r_tk

        g_pil = Image.fromarray(g)
        g_tk = ImageTk.PhotoImage(g_pil)
        self.g_label.config(image=g_tk)
        self.g_label.image = g_tk

        b_pil = Image.fromarray(b)
        b_tk = ImageTk.PhotoImage(b_pil)
        self.b_label.config(image=b_tk)
        self.b_label.image = b_tk

    # Hàm để chuyển ảnh thành ảnh đa mức xám (gray scale) và hiển thị
    def convert_to_gray(self):
        # Kiểm tra xem đã chọn ảnh chưa
        if self.original_image is None:
            return

        # Lấy ảnh gốc từ biến instance
        original_image = self.original_image

        # Chuyển ảnh sang ảnh đa mức xám
        gray_image = cv2.cvtColor(original_image, cv2.COLOR_RGB2GRAY)

        # Hiển thị ảnh đa mức xám trong label
        gray_image_pil = Image.fromarray(gray_image)
        gray_image_tk = ImageTk.PhotoImage(gray_image_pil)
        self.original_image_label.config(image=gray_image_tk)
        self.original_image_label.image = gray_image_tk

    # Hàm để xóa toàn bộ ảnh đã hiển thị
    def reset_images(self):
        # Kiểm tra xem đang trong quá trình quay ảnh hay không
        if self.is_rotating == False:
            # self.rotation_process.join()  # Đợi tiến trình quay ảnh kết thúc trước
khi tiếp tục
            self.original_image_label.config(image=None)
            self.original_image_label.image = None
```

```
        # Xóa ảnh gốc
        self.original_image_label.config(image=None)
        self.original_image_label.image = None

        # Xóa các lớp R, G, B
        self.r_label.config(image=None)
        self.r_label.image = None

        self.g_label.config(image=None)
        self.g_label.image = None

        self.b_label.config(image=None)
        self.b_label.image = None

    # Hàm để quay ảnh
    def rotate_images(self):
        # Kiểm tra xem đã chọn ảnh chưa
        if self.original_image is None:
            return

        # Kiểm tra xem đang trong quá trình quay ảnh hay không
        if self.rotation_process and self.rotation_process.is_alive():
            return

        # Đặt biến cờ báo hiệu đang trong quá trình quay ảnh
        self.is_rotating = True

        # Lấy ảnh gốc từ biến instance
        original_image = self.original_image.copy()

        # Bắt đầu tiến trình quay ảnh
        self.rotation_process = Thread(target=self.rotate_images_thread,
args=(original_image,))
        self.rotation_process.start()

    # Hàm chạy trong tiến trình để quay ảnh
    def rotate_images_thread(self, original_image):
        # Thực hiện quay ảnh 100 lần, mỗi lần quay 10 độ và tạm dừng 0.1 giây
        for _ in range(10):
            original_image = self.rotate_image(original_image, angle=10)
            self.display_rotated_image(original_image)
            self.master.update()
            time.sleep(0.1)

        # Đặt biến cờ trạng thái quay ảnh về False sau khi kết thúc
        self.is_rotating = False
```

```
    # Hàm để quay ảnh một góc nhất định
    def rotate_image(self, image, angle):
        rows, cols, _ = image.shape
        M = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, 1)
        rotated_image = cv2.warpAffine(image, M, (cols, rows),
borderMode=cv2.BORDER_CONSTANT, borderValue=(255, 255, 255))
        return rotated_image
```

```
    # Hàm để hiển thị ảnh sau khi quay
    def display_rotated_image(self, image):
        # Hiển thị ảnh gốc trong label
        rotated_image_pil = Image.fromarray(image)
        rotated_image_tk = ImageTk.PhotoImage(rotated_image_pil)
        self.original_image_label.config(image=rotated_image_tk)
        self.original_image_label.image = rotated_image_tk
```

```
    # Hàm để cắt ảnh
    def crop_image(self):
        # Kiểm tra xem đã chọn ảnh chưa
        if self.original_image is None:
            return
```

```
        # Lấy ảnh gốc từ biến instance
        original_image = self.original_image
```

```
        # Lấy kích thước ảnh gốc
        height, width, _ = original_image.shape
```

```
        # Tính toán kích thước cần cắt (1/4 ảnh tính từ tâm ảnh)
        new_width = int(width / 2)
        new_height = int(height / 2)
```

```
        # Tính toán vị trí bắt đầu của ảnh cắt
        start_x = int(width / 4)
        start_y = int(height / 4)
```

```
        # Cắt ảnh
        cropped_image = original_image[start_y:start_y + new_height,
start_x:start_x + new_width]
```

```
        # Hiển thị ảnh trước khi cắt
        original_image_pil = Image.fromarray(original_image)
        original_image_tk = ImageTk.PhotoImage(original_image_pil)
        self.original_image_label.config(image=original_image_tk)
        self.original_image_label.image = original_image_tk
```

```
        # Hiển thị ảnh sau khi cắt
        cropped_image_pil = Image.fromarray(cropped_image)
        cropped_image_tk = ImageTk.PhotoImage(cropped_image_pil)
        self.r_label.config(image=cropped_image_tk)
        self.r_label.image = cropped_image_tk

# Tạo cửa sổ tkinter
root = tk.Tk()

# Tạo đối tượng ứng dụng
app = ImageSelectorApp(root)

# Main loop của cửa sổ tkinter
root.mainloop()
```