You are provided with a header file and source code capable of creating, loading and saving a bitmap image in bmp format. Your task is to implement a utility to perform image manipulation. The utility should be able to select a given area of an image, resize it, and place it in the original image.

The **first** command line argument of your programme will be the file name of the input image, the **second** command line argument will be the name of the output image. The **rest** of the arguments will contain cropping information in the following format:
crop source_top_left_x source_top_left_y source_bottom_right_x source_bottom_right_y destination_top_left_x destination_top_left_y destination_bottom_right_x destination_bottom_right_y.

Your program should be able to process several crop operations in the sequence they are provided on the command line. When resizing, do not worry about pixel colour interpolation. If scaling up, use the colour of the neighbouring pixels multiple times, i.e., copy a given pixel several times. When scaling down, leave out some pixels to achieve the desired reduction in scale.

The top left pixel has coordinates 0 0. If the provided coordinates are out of the image boundaries, report an error. Input and output images have the same size. In case there is no crop performed, the output image should be identical to the input one. Assume that the source and destination areas do not overlap.

Image 1

Image 2

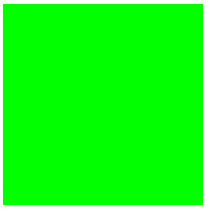

Image 3

Example: Assume that image 1 is 1000 x 1000 pixels. Then, "**./main image_1.bmp image_2.bmp crop 0 0 99 99 100 400 900 600 crop 0 900 99 999 400 100 600 900**" should convert image 1 to image 2 or "**./main image_1.bmp image_2.bmp crop 0 0 99 99 0 0 999 999**" should convert image 1 to image 3.

Your program should be able to handle the following line of arguments: "**image_1.bmp image_2.bmp crop 0 0 1 1 100 0 150 400 crop 0 0 1 1 150 100 200 200 crop 0 0 1 1 200 200 250 300 crop 0 0 1 1 250 0 300 400 crop 0 0 1 1 350 0 400 400 crop 0 0 1 1 450 0 500 400 crop 0 0 1 1 500 0 600 50 crop 0 0 1 1 500 350 600 400 crop 450 0 600 400 650 0 800 400 crop 0 0 1 1 700 180 750 220 crop 0 0 1 1 50 500 100 900 crop 0 0 1 1 0 500 150 550 crop 0 0 1 1 200 550 250 850 crop 0 0 1 1 350 550 400 850 crop 0 0 1 1 250 500 350 550 crop 0 0 1 1 250 850 350 900 crop 450 0 600 400**

530 500 680 900 crop 0 0 1 1 800 500 850 850 crop 0 0 1 1 850 850 950 900 crop 0 0 1 1 950 500 990 850 crop 200 950 200 950 0 0 99 99 crop 200 950 200 950 900 0 999 99 crop 200 950 200 950 0 899 999 999″

The C99 documentation and lecture slides are available to you on the course wiki. The C99 standard is strictly required.

You have to zip and upload your solution to the brute system.
Suggested but optional walkthrough and how we will evaluate in terms of points :

1) Download and unzip the assignment files (e.g. by "**wget https://cw.fel.cvut.cz/wiki/_media/courses/be5b99cpl/cpl_2024_01_27.zip"**). Check their structure and create a compilable main.c, which reports if the command line arguments are not in the expected format. **(1 point)**

2) Write a Makefile that compiles and links the image (simage.cpp, simage.h) module. **(1 point)**

3) Modify your main.c so that it can load and save an image using the provided 'simage.h' and 'simage.cpp' codes. Check the saved image. **(1 point)**

4) Report if the crop command coordinates are out of image boundaries. **(1 point)**

5) Implement the crop functionality without resizing. **(2 points)**

6) Enhance the crop functionality with resizing. **(3 points)**

7) Make sure that all your code functions have comments regarding their arguments and describing what the function does. Correct any compiler warnings/errors when compiling with "`-Wall -Werror -std=c99 -pedantic`" flags. **(1 point if 1-5 are implemented)**