Lab 5 - Report

Neagu Lorena

1. For the first requirement, I implemented the first algorithm for connected-component labelling with breadth first traversal as described in the lab. I used a pre-implemented queue for the BF traversal and i used two functions for computing the 4 neighbors or the 8 neighbors of a pixel. In the function call, the user can easily choose the neighboring type by specifying in nbh\_type parameter of the function CCLabelling. In case the user inputs nbh\_type any other value than 4/8, he will be notified.

Here at first i accesed the neighboring positions wrong, instead of accesing the coordinates as {neighbors[k].x, neighbors[k].y} I used neighbors[k] and so i got an error. I fixed that and it works now.

1. For the generation of the colored image based on the labels computed, i assigned for each label a randomly generated color, and updated the initial image. The initial image and the resulted images are shown as output.

Here we had to generate the image by assigning to each base color RGB a randomly generated number from 0 to 255. So we needed 3 random numbers for each color, firstly i used only 1, but corrected it afterwards.

1. For the second algorithm, I also used the code from the laboratory for the implementation of the two pass labelling.

I had to use “std::vector<int> edges[1000];” ,instead of “v**ector<vector<int>> edges;”**

, as it was written in the laboratory, because i had a “segmentation fault error” or the execution stopped unexpectedly. I had to assume there are no more than 1000 labels.

The result is showed as output to the screen. Both the result after the first and the second pass are showed such that the two can be compared. For the first result, after the first pass, I used the labels matrix from the first pass to generate a colored image, like i used for the second and final result as well. Also, i showed in the console the number of labels after the first and the second