proj1b.cpp

```
#include "opencv2/highgui.hpp"
#include <iostream>
using namespace cv;
using namespace std;

void runOnWindow(int W1,int H1, int W2,int H2, Mat inputImage, char *outName) {
  int rows = inputImage.rows;
  int cols = inputImage.cols;

  vector<Mat> i_planes;
  split(inputImage, i_planes);
  Mat iB = i_planes[0];
  Mat iG = i_planes[1];
  Mat iR = i_planes[2];

  // dynamically allocate RGB arrays of size rows x cols
  int** R = new int*[rows];
  int** G = new int*[rows];
  int** B = new int*[rows];
  for(int i = 0 ; i < rows ; i++) {
    R[i] = new int[cols];
    G[i] = new int[cols];
    B[i] = new int[cols];
  }

  for(int i = 0 ; i < rows ; i++)
    for(int j = 0 ; j < cols ; j++) {
      R[i][j] = iR.at<uchar>(i,j);
      G[i][j] = iG.at<uchar>(i,j);
      B[i][j] = iB.at<uchar>(i,j);
    }


  //       The transformation should be based on the
  //       historgram of the pixels in the W1,W2,H1,H2 range.
  //       The following code goes over these pixels

  for(int i = H1 ; i <= H2 ; i++)
    for(int j = W1 ; j <= W2 ; j++) {
      double r = R[i][j];
      double g = G[i][j];
      double b = B[i][j];
      int gray = (int) (0.3*r + 0.6*g + 0.1*b + 0.5);

      R[i][j] = G[i][j] = B[i][j] = gray;
    }

  Mat oR(rows, cols, CV_8UC1);
  Mat oG(rows, cols, CV_8UC1);
  Mat oB(rows, cols, CV_8UC1);
  for(int i = 0 ; i < rows ; i++)
    for(int j = 0 ; j < cols ; j++) {
      oR.at<uchar>(i,j) = R[i][j];;
      oG.at<uchar>(i,j) = G[i][j];;
      oB.at<uchar>(i,j) = B[i][j];;
    }

  Mat o_planes[] = {oB, oG, oR};
  Mat outImage;
  merge(o_planes, 3, outImage);

  namedWindow("output", CV_WINDOW_AUTOSIZE);
  imshow("output", outImage);
  imwrite(outName, outImage);
}

```

```
67  int main(int argc, char** argv) {
68    if(argc != 7) {
69      cerr << argv[0] << ": "
70           << "got " << argc-1
71           << " arguments. Expecting six: w1 h1 w2 h2 ImageIn ImageOut."
72           << endl ;
73      cerr << "Example: proj1b 0.2 0.1 0.8 0.5 fruits.jpg out.bmp" << endl;
74      return(-1);
75    }
76    double w1 = atof(argv[1]);
77    double h1 = atof(argv[2]);
78    double w2 = atof(argv[3]);
79    double h2 = atof(argv[4]);
80    char *inputName = argv[5];
81    char *outputName = argv[6];
82
83    if(w1<0 || h1<0 || w2<=w1 || h2<=h1 || w2>1 || h2>1) {
84      cerr << " arguments must satisfy 0 <= w1 < w2 <= 1"
85           << ",  0 <= h1 < h2 <= 1" << endl;
86      return(-1);
87    }
88
89    Mat inputImage = imread(inputName, CV_LOAD_IMAGE_UNCHANGED);
90    if(inputImage.empty()) {
91      cout <<  "Could not open or find the image " << inputName << endl;
92      return(-1);
93    }
94
95    string windowInput("input: ");
96    windowInput += inputName;
97
98    namedWindow(windowInput, CV_WINDOW_AUTOSIZE);
99    imshow(windowInput, inputImage);
100
101   if(inputImage.type() != CV_8UC3) {
102     cout <<  inputName << " is not a standard color image  " << endl;
103     return(-1);
104   }
105
106   int rows = inputImage.rows;
107   int cols = inputImage.cols;
108   int W1 = (int) (w1*(cols-1));
109   int H1 = (int) (h1*(rows-1));
110   int W2 = (int) (w2*(cols-1));
111   int H2 = (int) (h2*(rows-1));
112
113   runOnWindow(W1, H1, W2, H2, inputImage, outputName);
114
115   waitKey(0); // Wait for a keystroke
116   return(0);
117 }
```