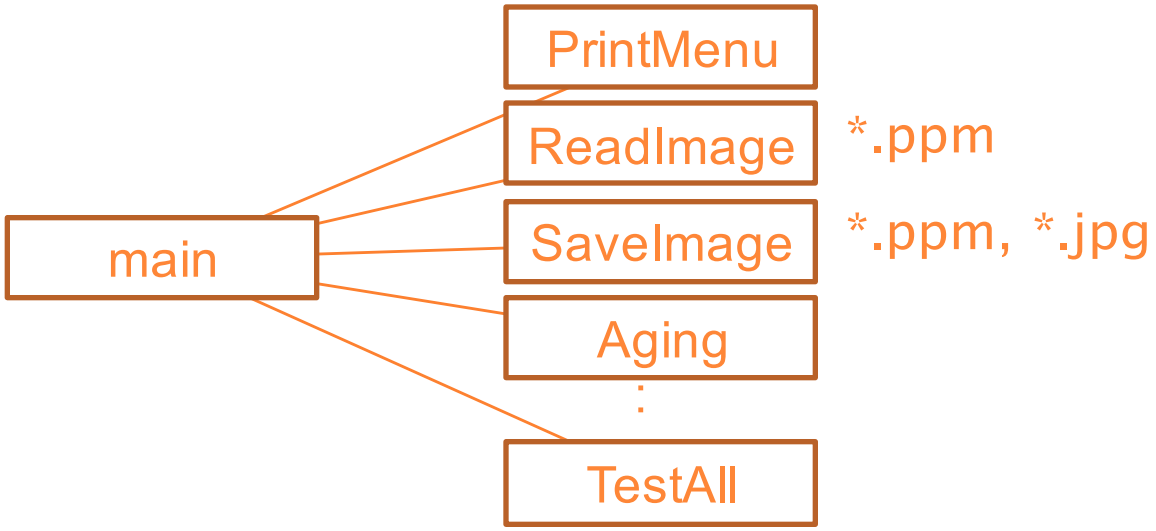
EECS 22: Advanced C Programming
Assignment 2

Huan Chen
huanc3@uci.edu

10/7/2016

Assignment 2

- A menu driven digital image processing program [100 pts]
- **Deadline : 2016/10/20, Thursday, 6:00 pm**
- Goal
  - Main function use function calls to input/output image, process image, and test all of the digital image process functions.



Menu Driven Digital Image Processing

```
eecs22@zuma.eecs.uci.edu:6 > ./PhotoLab
--------------------------------
 1:  Load a PPM image
 2:  Save an image in PPM and JPEG format
 3:  Make a negative of an image
 4:  Color filter an image
 5:  Sketch the edge of an image
 6:  Flip an image horizontally
 7:  Mirror an image vertically
 8:  Add Border to an image
 9:  Zoom an image
10:  Test all functions
11:  Exit
please make your choice:
```

Input File

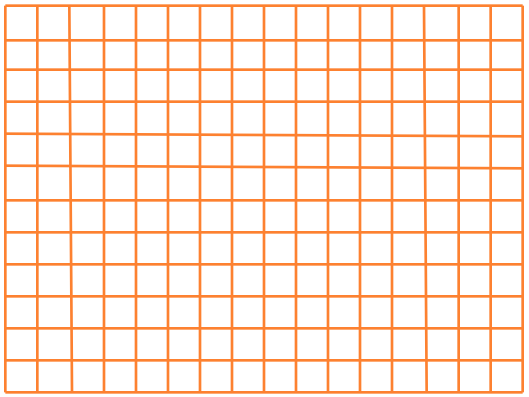
- Format : ppm
- Option 1: input ppm file
  - `Load a PPM image`
  - example 1:
    - ```
      please make your choice: 1
      Please input the file name to load: EH
      EH.ppm was read successfully!
      ```
  - File extension is not needed.
  - example 2:
    - ```
      please make your choice: 1
      Please input the file name to load: EH.ppm
      Cannot open file ”EH.ppm.ppm" for reading!
      ```
  - Function for reading image `ReadImage` is provided !

Output File

- Format : ppm, jpg
- Option 2: output ppm and jpg files
  - `Save an image in PPM and JPEG format`
  - example:
    - `Please make your choice: 2`
    - `Please input the file name to save: negative`
    - `negative.ppm was saved successfully.`
    - `negative.jpg was stored for viewing.`
  - File extension is not needed.
  - Function for saving image `SaveImage` is provided !

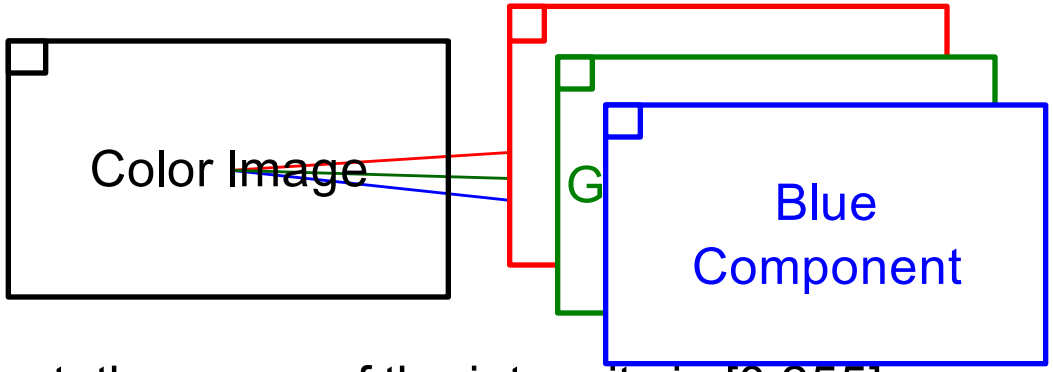
Picture in the program

- How to represent a picture in computer?
  - A picture is composed of pixels
  - One color for each pixel
  - Example: 16x12 = 192 pixels



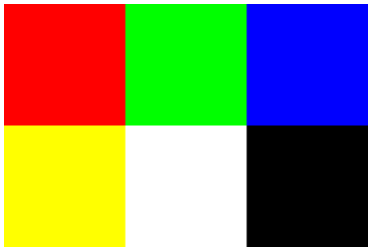
Picture in the program

- 3-tuple (R, G, B)
  - R: intensity of Red
  - G: intensity of Green
  - B: intensity of Blue
  - For image in ppm format, the range of the intensity is [0,255], using unsigned char for each intensity



- Color examples:
  - Red (255, 0, 0), Green (0, 255, 0), Blue (0, 0, 255)
  - Yellow (255, 255, 0), Cyan (0, 255, 255), Magenta(255, 0, 255)
  - White (255, 255, 255), black(0, 0, 0)
- PPM example
- RGBRGBRGBRGB…
-

```
P3    (colors)
3 2   (3 columns, 2 rows)
255   (255 for max color)
255   0 0       0 255   0     0 0 255
255 255 0     255 255 255     0 0   0
```

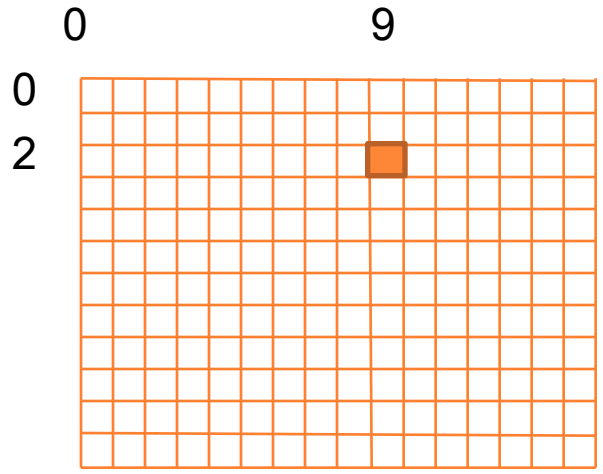


Picture in the program

- The data structure to represent a picture in this assignment
  - Two-dimensional arrays for the intensities of each pixel
    - For an image of size 16x12…

```
unsigned char R[16][12];
unsigned char G[16][12];
unsigned char B[16][12];
```

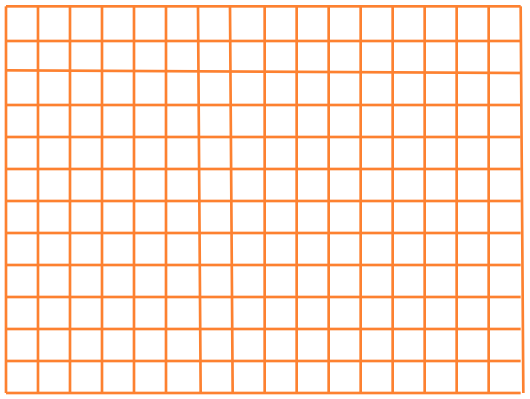
    - How to access a pixel in an image?
      - Coordinate of a pixel (x, y)
      - x = number of the column
      - y = number of the row
      - The color of the pixel (x, y) = (R[x][y], G[x][y], B[x][y])



Picture in the program

- How to access every pixel in the picture?
  - List all possible coordinates of the pixel
  - Two for-loops to scan all the pixels in a 2-D array
  - Inner loop
    - fix the number of the column, iterate the pixel in the same column with different row numbers
  - Outer loop
    - iterate all the columns

```
int x, y ;
for (x=0; x < 16; x++){
 for (y=0; y < 12; y++){
  processing on pixel(x, y);
 }
}
```



Digital Image Processing Function

```
eecs22@zuma.eecs.uci.edu:6  > ./PhotoLab
--------------------------------
 1:  Load a PPM image
 2:  Save an image in PPM and JPEG format
 3:  Make a negative of an image
 4:  Color filter an image
 5:  Sketch the edge of an image
 6:  Flip an image horizontally
 7:  Mirror an image vertically
 8:  Add Border to an image
 9:  Zoom an image
10:  Test all functions
11:  Exit
please make your choice:
```

- Note: Your program should response “Image is not in the program yet” if the user want to choose option 3~9 before using option 1 to read the image.
- If user inputs a invalid number (like 12), you should print error prompt message

Negative

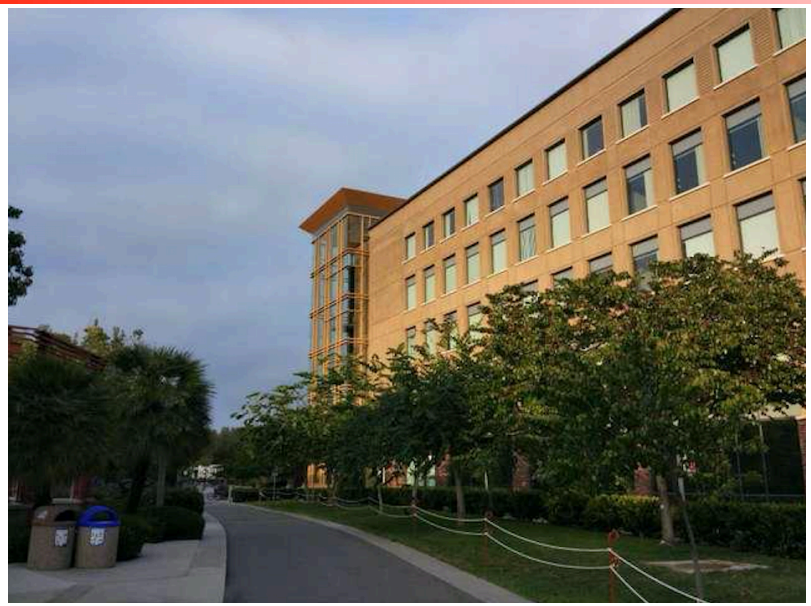
- Pseudo Code:
  For all pixels in the picture, subtract 255 which is the maximum intensity

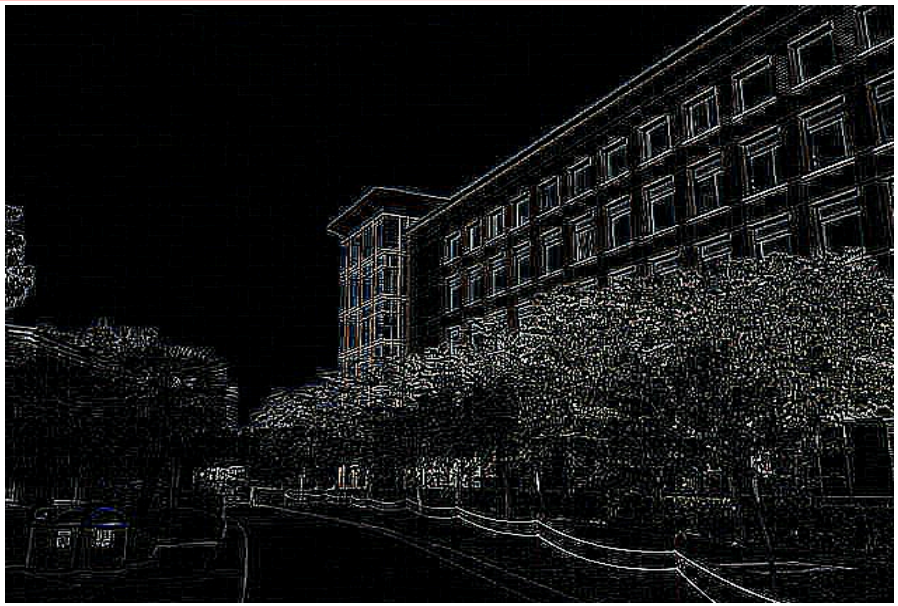
Color Filter

- For all pixels in the picture
  if (R in the range of [target_r – threshold, target_r + threshold]) and
    (G in the range of [target_g – threshold, target_g + threshold]) and
    (B in the range of [target_b – threshold, target_b + threshold])
      R = replace_r ;
      G = replace_g ;
      B = replace_b ;
  else
      keep the current color

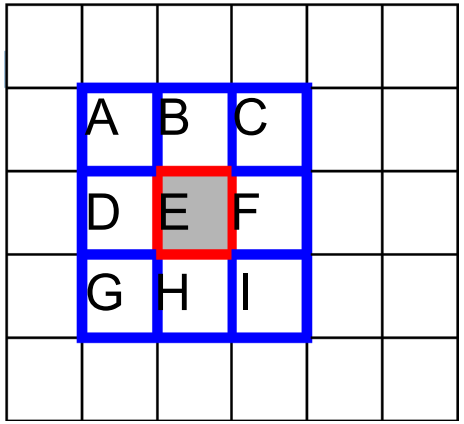
target_r = 190 replace_r = 0
target_g = 100 replace_g = 0
target_b = 150 replace_b = 255
Threshold = 60

Edge

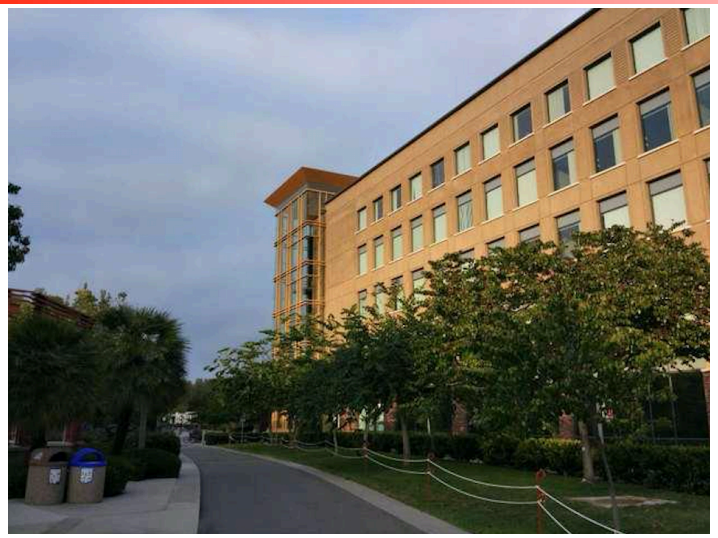


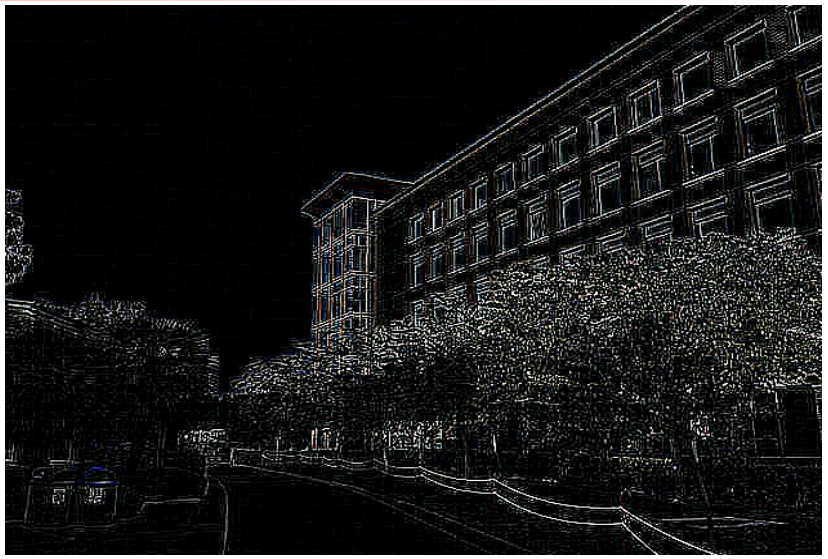


- Set the pixel's color at E with equation:
  new_E = 8*E – A – B – C – D – F – G – H – I
- Use temporary array to avoid computing with contaminated color intensities.
- Pixels on the corners and the edges have fewer neighbors.
- new_E should be in the range [0, 255]



Edge





For simplicity (without considering border pixels with less than 8 neighbors), you only process rows [1, HEIGHT – 2], cols [1, WIDTH - 2] in original image and set boundaries of output image to be zero

300 = (70 -10) + (70 - 20) + (70 – 30)
+ (70 - 60) + (70 – 80)
+ (70 -10) + (70 - 20) + (70 – 30)

a) Original

10	20	30	40	50
60	70	80	90	100
10	20	30	40	50
60	70	80	90	100

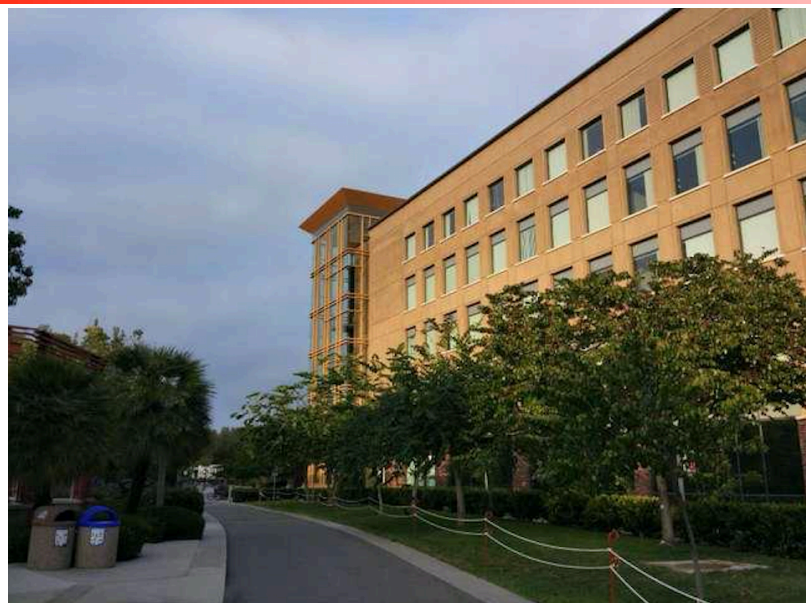
b) Ouput with Zero borders

0	0	0	0	0
0	300	..	..	0
0	..	..	..	0
0	0	0	0	0

Handle under- or overflow

0	0	0	0	0
0	255	..	..	0
0	..	..	..	0
0	0	0	0	0

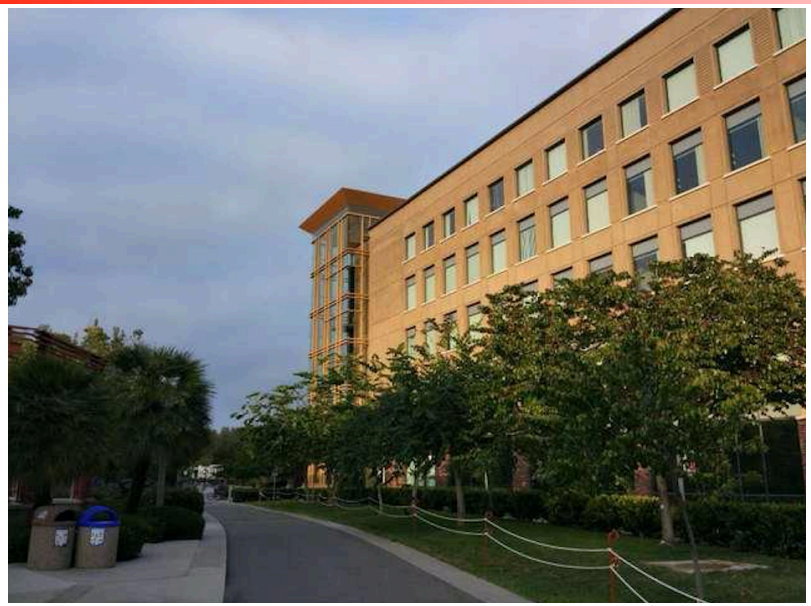
Horizontally Flip

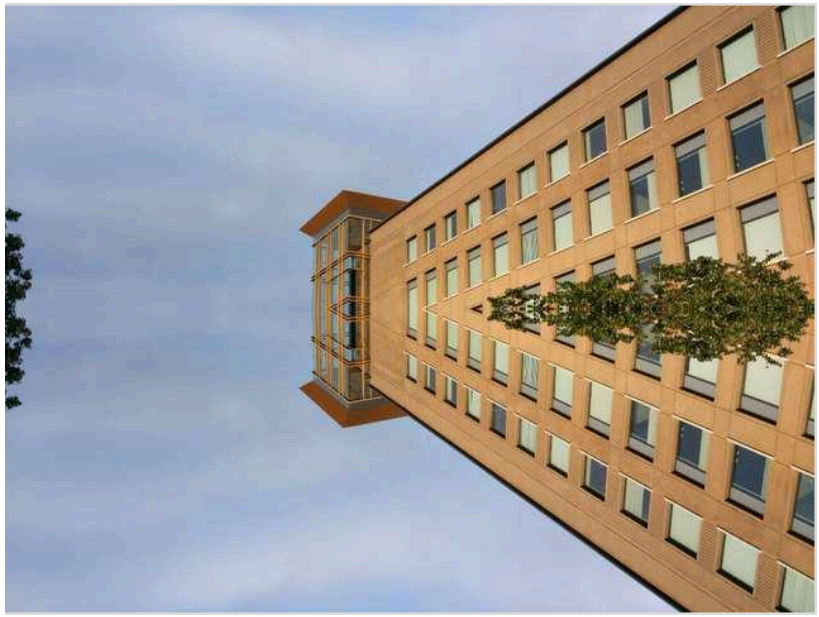


- For all pixels in the left half picture, swap the color with the pixel in the right half

```
1 2 3 4 5    5 4 5 2 1
0 1 2 3 4    4 3 2 1 0
3 4 5 6 7    7 6 3 4 3
```

Vertically Mirror





- For all pixels in the bottom half of the picture, replace the color to the color of pixel in the top half.

1 4 6	1 4 6
5 2 1	5 2 1
7 8 9	7 8 9
8 2 4	5 2 1
9 3 7	1 4 6

Add Border

- `void AddBorder(unsigned char R[WIDTH][HEIGHT],`
  `unsigned char G[WIDTH][HEIGHT],`
- `unsigned char B[WIDTH][HEIGHT],`
- `int r, int g, int b, int bwidth);`

  Define an aspect ratio of 16:9 (horizontal border thicker)

Add Border

- `int strcmp(const char *str1, const char *str2)`
  - Used to compare two character strings. Returns 0 if they are the same.
- `Border colors:`
  - Red (255, 0, 0), Green (0, 255, 0), Blue (0, 0, 255)
  - Yellow (255, 255, 0), Cyan (0, 255, 255), Pink (255, 192, 203)
  - Orange (255, 165, 0), White (255, 255, 255), Black(0, 0, 0)

Zoom

a) Original

10	20	30	40	50
60	70	80	90	100
10	20	30	40	50
60	70	80	90	100

b) Zoom (x-direction)

10	15	20	25	30	35	40	45	50
60	65	70	75	80	85	90	95	100
10	15	20	25	30	35	40	45	50
60	65	70	75	80	85	90	95	100

Zoom

c) Zoom (y-direction)

d) Final

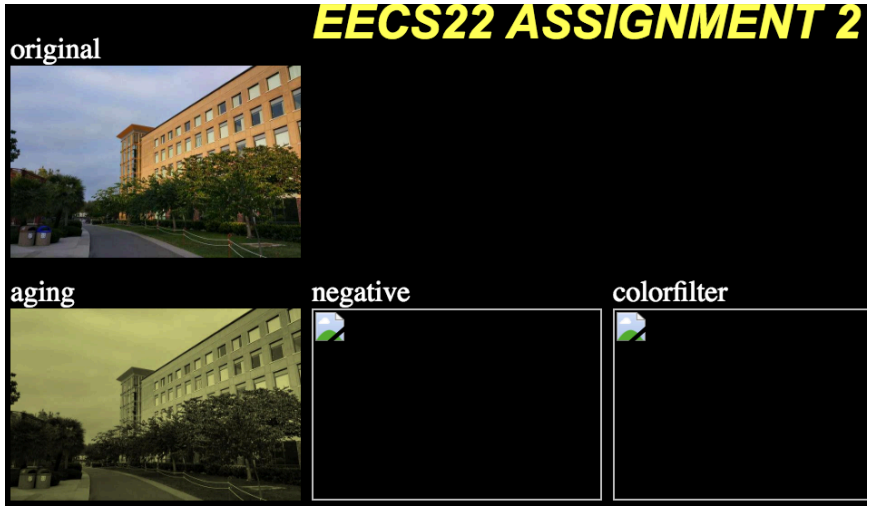
10	15	20	25	30	35	40	45	50
35	40	45	50	55	60	65	70	75
60	65	70	75	80	85	90	95	100
35	40	45	50	55	60	65	70	75
10	15	20	25	30	35	40	45	50
35	40	45	50	55	60	65	70	75
60	65	70	75	80	85	90	95	100

You should pick the center block from step c) as the output of ZOOM

70	75	80	85	90
45	50	55	60	65
20	25	30	35	40
45	50	55	60	65

Initial Setup

1. Fetch hw2 template code and source image
   - `mkdir hw2`
   - `cd hw2`
   - `cp ~eecs22/public/PhotoLab.c .`
   - `cp ~eecs22/public/EH.ppm .`
2. Compile and run hw2 template code, which generates a webpage available at https://newport.eecs.uci.edu/~YOUR_UCI_NET_ID
   - `gcc -Wall -ansi -o PhotoLab PhotoLab.c`
   - `./PhotoLab`
3. Go to https://newport.eecs.uci.edu/~YOUR_UCI_NET_ID, check it out!



Provided Function

```
const int WIDTH = 640;        /* Image width */
const int HEIGHT = 480;       /* image height */
#define SLEN    100       /* maximum length of file names */
#define ZOOM_FACTOR   2   /* Zooming factor for the zoom function */

int main()
{
 /*
   * Two dimensional arrays to hold the current image data
   * One array for each color component
   */
     unsigned char   R[WIDTH][HEIGHT];
     unsigned char   G[WIDTH][HEIGHT];
     unsigned char   B[WIDTH][HEIGHT];
/*  Please replace the following code with proper menu  */
/*  with function calls for DIP operations              */
     AutoTest(R, G, B);
/*  end of replacing*/
     return 0;
}
```

Provided Function

- Image Input / Output
  - ```
    int ReadImage (char fname[SLEN],
                   unsigned char R[WIDTH][HEIGHT],
                   unsigned char G[WIDTH][HEIGHT],
                   unsigned char B[WIDTH][HEIGHT]) ;
    ```
  - ```
    int SaveImage (char fname[SLEN],
                   unsigned char R[WIDTH][HEIGHT],
                   unsigned char G[WIDTH][HEIGHT],
                   unsigned char B[WIDTH][HEIGHT]) ;
    ```
  - Arguments are passed to the function by reference.
- Use `scanf("%s", fname)` to input file name

Provided Function

- Aging function – as the sample of DIP function
  - ```
    void Aging(unsigned char R[WIDTH][HEIGHT],
               unsigned char G[WIDTH][HEIGHT],
               unsigned char B[WIDTH][HEIGHT])
    {
      int x, y;
      for( y = 0; y < HEIGHT; y++ )
        for( x = 0; x < WIDTH; x++ ) {
          B[x][y] = ( R[x][y]+G[x][y]+B[x][y] )/5;
          R[x][y] = (unsigned char) (B[x][y]*1.6);
          G[x][y] = (unsigned char) (B[x][y]*1.6);
      }
    }
    ```
    ```