EECS 22 : Assignment 4

DIGITAL IMAGE PROCESSING

DUE DATE: 11/25/2014 11:00PM

https://eee.uci.edu/14f/18030/

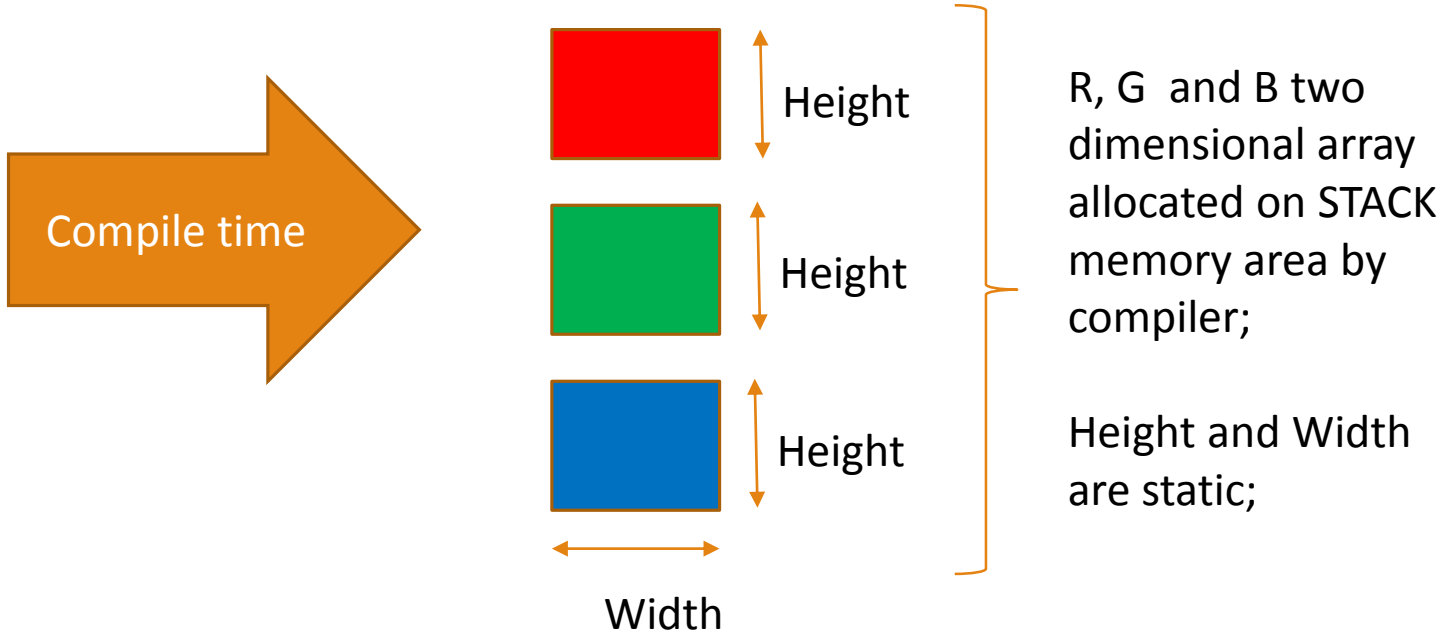
Presented by : Manjunath M Venkatesh

Static Memory Allocation

Memory allocation for image in project 3

```
Void int main()
{
        unsigned char R[WIDTH][HEIGHT];
        unsigned char G[WIDTH][HEIGHT];
        unsigned char B[WIDTH][HEIGHT];
                .
                .
                .
                .


}
```

- Need to know WIDTH and HEIGHT before compiling code
- WIDTH and HEIGH are static ! Cannot be change on runtime (after compilation)



Limitations

Image size should be known before at compile time.

Not any image can be loaded for DIP operation.

Solution : Dynamic memory allocation

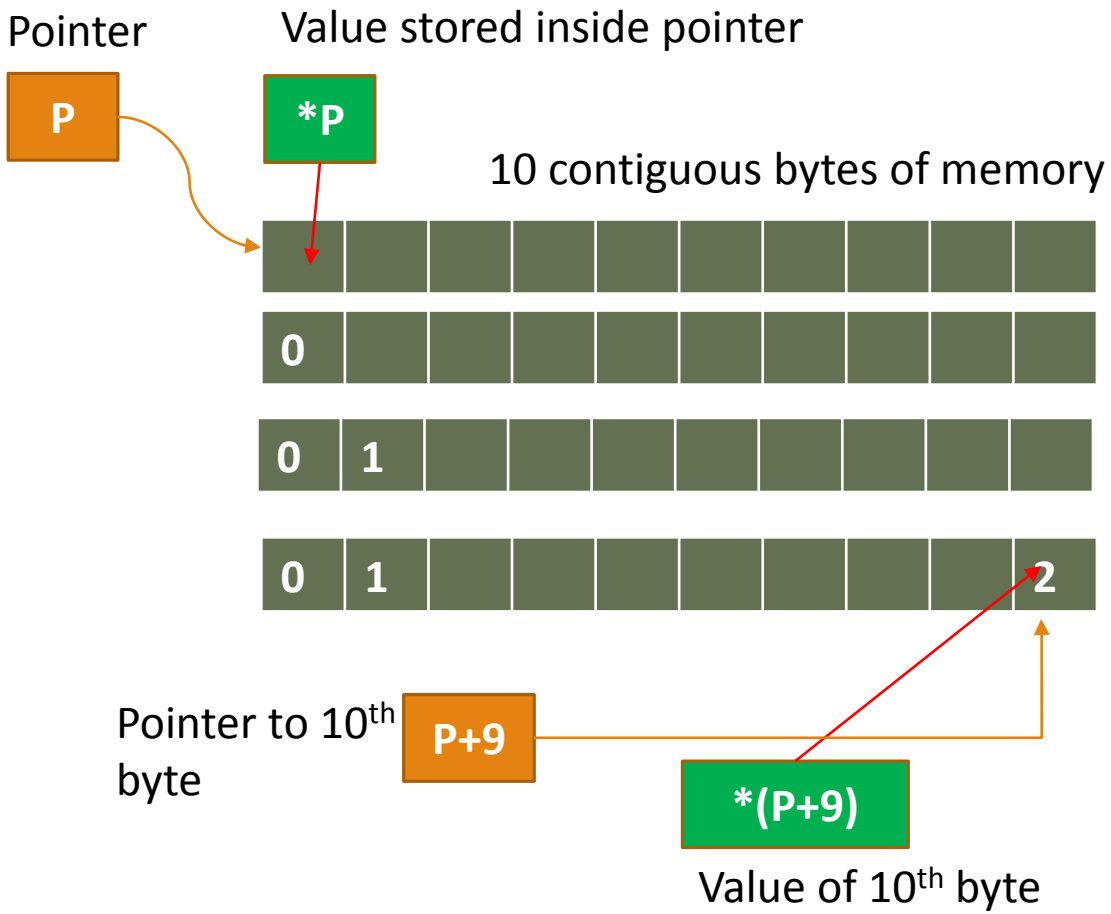
Allocate memory at runtime.

Need not know actual dimension at time of writing and compiling code.

Dynamic Memory Allocation

- **Malloc** API allocates X number of bytes on HEAP memory at runtime, where X is passed as input parameter and returns pointer to allocated memory

```
Void int main()
{
        char temp = 0;
        unsigned char p* = Null;

        p= malloc(10);

        *p = 0;  //same as *(p+0) = 0

        *(p+1) = 1; // 2nd Byte

        *(p+9) = 2;     // 10th Byte

        temp = *(p+9); // temp will have 2 in it

        temp = (p+9);  // temp will have address
to 10th memory byte

        print("%c",*(p+9)); // will print 2;
}
```



Dynamic allocation of image data

Memory allocation in Project 4

```
Void int main()
{
        unsigned char *R;
        unsigned char *G;
        unsigned char *B;
        int WIDTH;
        int HEIGHT;
                .
        /* Get height and width from user
and store in WIDTH and HEIGHT, local
variable */

        R = (unsigned char *)malloc(WIDTH *
HEIGHT);
        G = (unsigned char *)malloc(WIDTH *
HEIGHT);
        B = (unsigned char *)malloc(WIDTH *
HEIGHT);
}
```

Runtime

- One dimensional memory

Pixel access in dynamic image data

Project 3: (with static allocation)

(x, y)th pixel can be accessed as below

```
temp = R[x][y];                    // two dimensional array
```

Project 4 : (with dynamic allocation)

- One dimensional byte access in case of dynamic memory allocated data
- Assuming row major data storage

Assume R is pointer to allocated memory then any pixel in image can be accessed by

temp = *(R+(x + (y * Width)))

Representing an Image

- **We need following information to represent an Image**

```
Unsigned int WIDTH;
Unsigned int HEIGHT;
Unsigned char *R;
Unsigned char *G;
Unsigned char *B;
```

Better way

Encapsulate common information inside c "Structure"

```
Struct IMAGE {
        Unsigned int WIDTH;
        Unsigned int HEIGHT;
        Unsigned char *R;
        Unsigned char *G;
        Unsigned char *B;
}
```

- **So new DIP functions should take following information as formal parameter**

- `Example:  BlackNWhite(WIDTH, HEIGHT, R, G, B);`

- **So new DIP functions shall take a pointer to structure of type Image**

- `Example:`

```
BlackNWhite(image);
```

Use this method for project 4

More information on “structure”

```
Struct IMAGE {
        Unsigned int WIDTH;
        Unsigned int HEIGHT;
        Unsigned char *R;
        Unsigned char *G;
        Unsigned char *B;
}
```

Example usage :

```
struct IMAGE *image; // defining pointer

image = (struct IMAGE *)malloc(sizeof(struct IMAGE));

Image->R = 0; // Accessing filed R of structure
```


```
Typedef struct{
        Unsigned int WIDTH;
        Unsigned int HEIGHT;
        Unsigned char *R;
        Unsigned char *G;
        Unsigned char *B;
} IMAGE;
```

Example usage :

```
IMAGE *image; // defining pointer

Image = (IMAGE *)malloc(sizeof(IMAGE));
Image->R = 0; // Accessing filed R of structure
```

Use this method for project 4
Refer Image.h file for more information

Note : -> : to access elements of a structure pointed by a pointer to that structure

sizeof(X) : Will return number bytes required to store element of type X

Freeing dynamic memory :

- Memory allocated using “**malloc**” should explicitly be freed using “**Free**”
- If not all memory is freed then application results in memory leak
  - I.e every time you run your application un freed memory will become un usable in the system.

Example :

```
char *p  = malloc(10); // allocation
free(p);      // freeing allocated memory
```

Implementation details :

Step 1 : Implement APIs specified in **image.h** file

```
GetPixelR
GetPixelG
GetPixelB
SetPixelR
SetPixelG
SetPixelB
CreateImage        //Return type is pointer to structure IMAGE
DeleteImage        // Should delete all dynamic memory
```

Note :

1. New FileIo.c file uses `CreateImage` API for its internal operation. Implement this carefully
2. Update **Makefile** to include **Image.c**
3. Hints for `CreateImage` :
   - `Allocate memory for image structure.`
   - `Allocate memory for R, G and B.`
   - `Store WIDTH and HEIGHT in structure;`
   - `Return pointer to image structure;`

Implementation details :

Step 2 : Update image processing functions to dynamically allocated memory format

example :

```
IMAGE *BlackNWhite(IMAGE *image);
IMAGE *HMirror(IMAGE *image);
IMAGE *AddBorder(IMAGE *image, char color[SLEN], int border_width);
      .
      .
      .
      .
      .
      .
```

Update “PhotoLab.c” and other functions accordingly

Implementation details :

Step 3 : New Advance DIP functions

1> Overlay function : (Update from Project 3)

Modify Overlay function to take overlay image of any size.

small image (144x168)

Small Image on RingMall

Implementation details :

2> Resize image : Enlarge or Shrink given image dimension

```
/*Resize*/
IMAGE *Resize(unsigned int percentage, IMAGE *image);
```

Widthnew = Widthold * (percentage / 100.00);

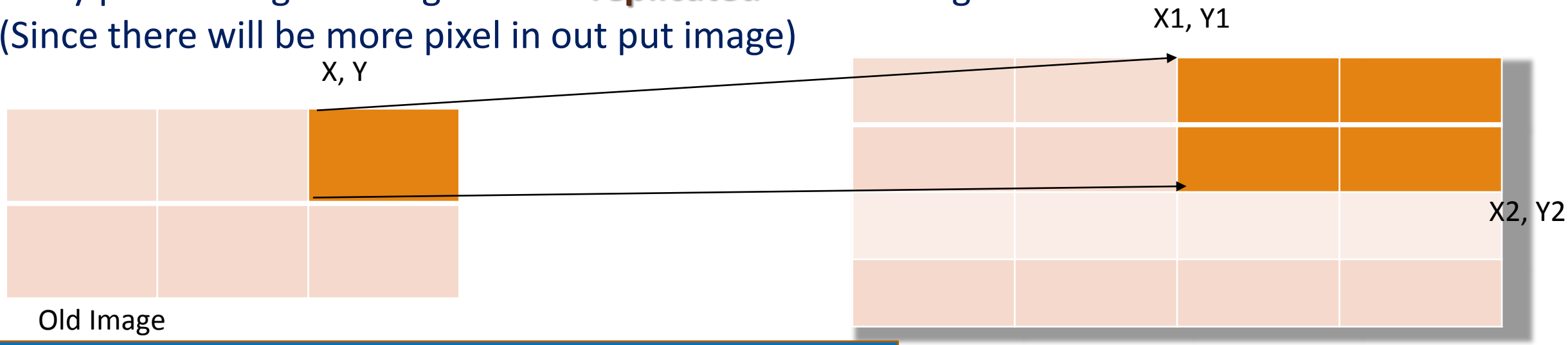
Heightnew = Heightold * (percentage / 100.00);

percentage == 100, the size of the new image is the same as the original one.
percentage < 100, the size of the new image is smaller than the original one.
percentage > 100, the size of the new image is larger than the original one.

Implementation details :

2> Resize image : Enlarge given image dimension

IF percentage is grater than **100** (Enlarge) : More pixels than the original image
Every pixel in original image will be **replicated** in new image
(Since there will be more pixel in out put image)

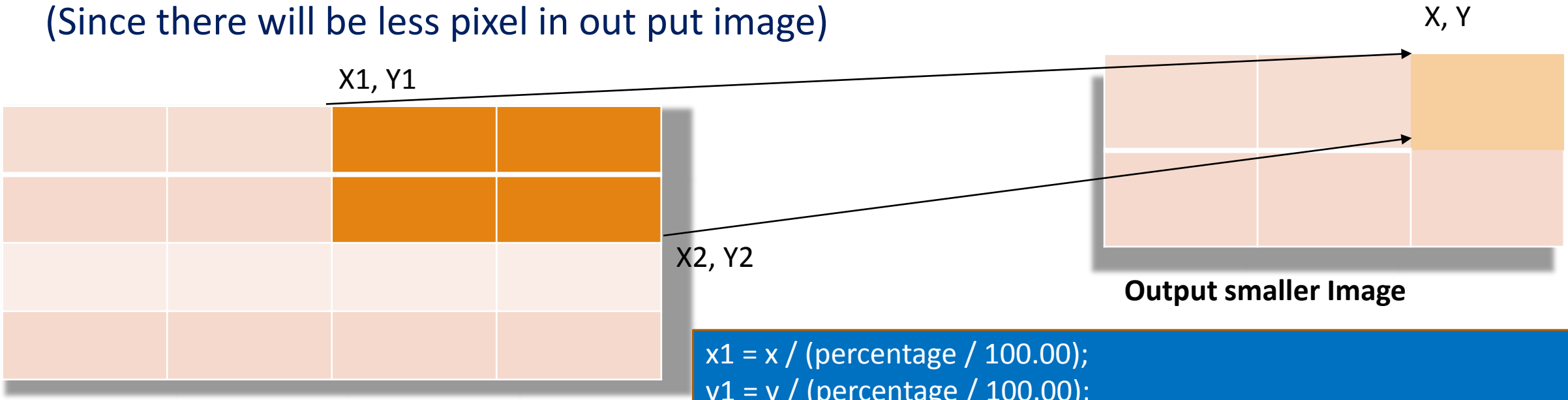



```
x1 = x / (percentage / 100.00);
y1 = y / (percentage / 100.00);
x2 = (x + 1) / (percentage / 100.00);
y2 = (y + 1) / (percentage / 100.00);
```

Implementation details :

2> Resize image : Shrink given image dimension

IF percentage is less than **100** (Enlarge) : Less pixels than the original image
Every pixel in output image will be **average** of few pixels in original image
(Since there will be less pixel in out put image)




```
x1 = x / (percentage / 100.00);
y1 = y / (percentage / 100.00);
x2 = (x + 1) / (percentage / 100.00);
y2 = (y + 1) / (percentage / 100.00);
```

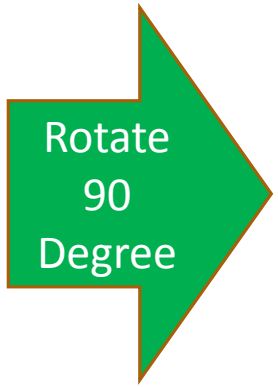
Implementation details :

3> Rotate image : Exchange Width and Height dimension of the given image

- Restore the image in dimension format

1	2	3	4	5	6
7	8	9	10	11	12

Input



7	1
8	2
9	3
10	4
11	5
12	6

Output

New Width = Old Height;
New Height = Old Width;

New image size == Old image size

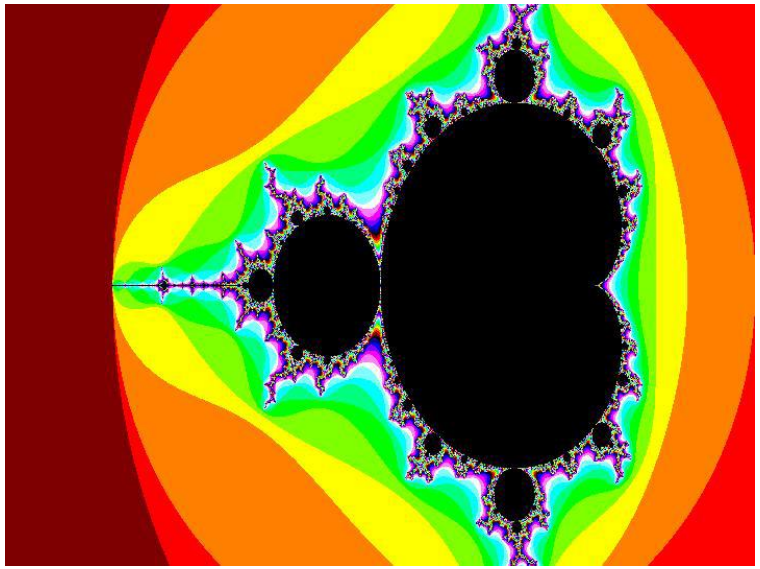
Implementation details :

3> Mandelbrot Set : (http://en.wikipedia.org/wiki/Mandelbrot_set)

- The algorithm computes the color for each pixels in the picture

based one their coordinates (Refer to assignment document for more information)

- **Pseudo** code for computing Mandelbrot set is given in assignment document.
  Your task is to write the given logic in “C” programing language.



Example output Image

Implementation details :

3> Mandelbrot Set : (http://en.wikipedia.org/wiki/Mandelbrot_set)

Hints :

a) Line 3 to Line 6 :

Max value of x0 is 1 and min value is -2.5;

```
X0 = (row /width * 3.5) – 2.5;    // Example mapping
```

Max value of y0 is 1 and min value is -1;

```
Y0 = (col /height * 2) – 1;     // Example mapping
```

b) Line 26 :

Assign RGB value indexed by color at pixel (row,col)

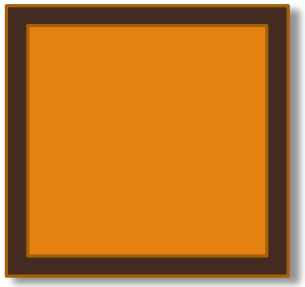
Implementation details :

4> **Bonus (10 points)** : External border on given image

```
IMAGE *AddOuterBorder(IMAGE *image, char color[SLEN], int
border_width)
```

The output image should be a bigger image containing border and the original image

Input Image
(WxL) size



Output Image
((W+(2*border_width)) x (L +(2*border_width)])) New size

User interface :

New user interface should look like

```
1: Load a PPM image
2: Save an image in PPM and JPEG format
3: Change a color image to Black & White
4: Flip an image vertically
5: Mirror an image horizontally
6: Color-Filter an image
7: Sketch the edge of an image
8: Shuffle an image
9: Posterize the image
10: Fill lights to an image
11: Overlay an image
12: Bonus, Cut and Paste operation on image
13: Resize the image
14: Rotate 90 degrees clockwise
15: Generate the Mandelbrot image
16: Bonus, Add border outside the image
17: Test all functions
18: Exit
Please make your choice:
```

Where to start :

1: Enter this command in home directory

```
cd ~/eecs22
mkdir hw4
cd hw4
cp /users/grad2/doemer/eecs22/hw4/RingMall.ppm .
cp /users/grad2/doemer/eecs22/hw4/Spider.ppm .
cp /users/grad2/doemer/eecs22/hw4/Peter.ppm .
cp /users/grad2/doemer/eecs22/hw4/FileIO.h .
cp /users/grad2/doemer/eecs22/hw4/FileIO.c .
cp /users/grad2/doemer/eecs22/hw4/Image.h .
```

2: Create **Image.c** file according to **Image.h** and modify **Makefile** accordingly

3: Use build commands as in project 3

4: Implement new DIP functions and update old DIP functions.

5: Update **PhotoLab.c** with new user interface and **Autotest** function

6: Test for memory leak using "**Valgrind**" (see assignment document section 1.7) (**Important !**)

Submission :

Use the standard submission procedure to submit the following files as the whole package of your program:

PhotoLab.c
PhotoLab.script
PhotoLab.txt
Image.c
Image.h
Constants.h
DIPs.c
DIPs.h
FileIO.c
FileIO.h
Advanced.c
Advanced.h
Makefile

Please leave the images generated by your program in your public html directory. Don't delete them as we may consider them when grading! You don't have to submit any images.

Note : You must implement "autotest" function testing all you DIP operation ! Other wise points will be deducted from your score.