EECS22 Lab Week5

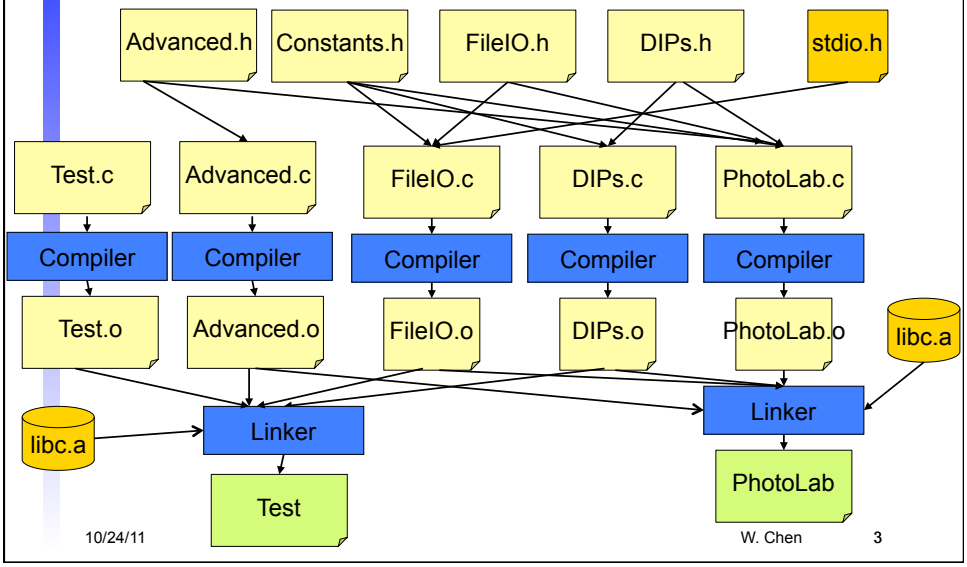
TA: Weiwei CHEN
Office hour: Mon, 11:00-12:50am EH 1141
weiwei.chen@uci.edu
eecs22@eecs.uci.edu

Assignment 3

- Decompose a program into multiple modules
- Compile multiple modules into different programs
- Makefile Development
- DEBUG mode support
- Advanced Digital Image Processing Functions
- Bitwise manipulations

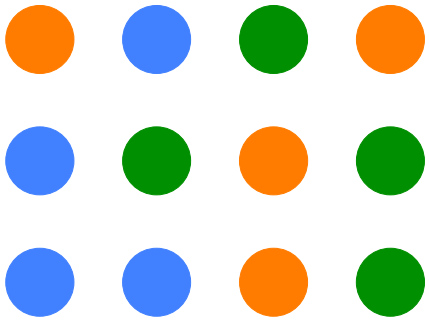
PhotoLab Modules

- **`PhotoLab (assignment2)`** decomposed into Modules



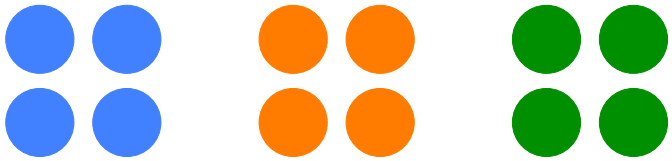
Jigsaw Activity on Assignment 3

Step1: Work on the question belongs to your color group individually. (5 minutes)



Jigsaw Activity on Assignment 3

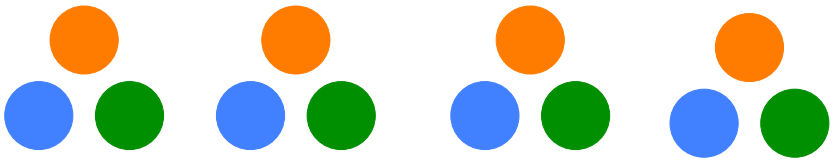
Step2: Share your answers with your classmates who are in the same color group. (4 minutes)



Jigsaw Activity on Assignment 3

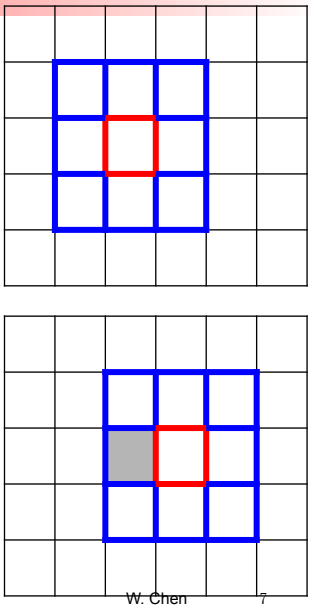
Step3: Now you are the expert on the question you have been working on. Please form into groups of 3, 1 blue, 1 orange, 1 green. Tell your group partner how to solve the question you've been working on!

(8 minutes)



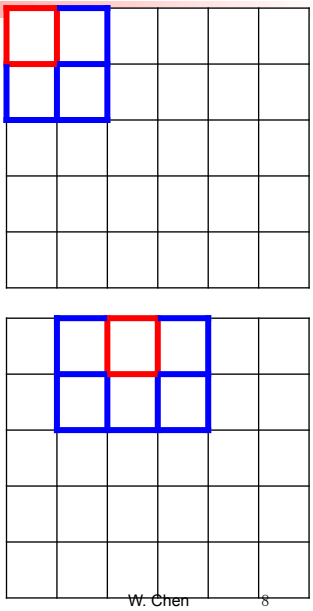
Advanced DIP functions

- Blurring
  - A pixel has 8 neighbors
  - Get the average values of the three channels of the current pixel and its 8 neighbors'.
  - Set the pixel's color components to the average values respectively.
  - In order not to contaminate the original value of the picture, use temporary arrays for computation and copy the result back to the original arrays.



Advanced DIP functions

- Blurring
  - Pixels on the corners and the edges.
  - Have fewer neighbors
  - Handle separately
  - Ignore pixels on the edges

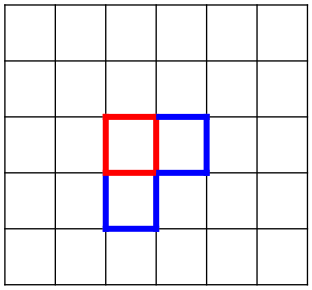


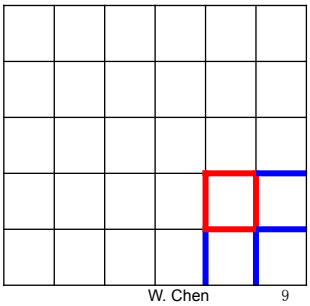
Advanced DIP functions

- Edge Detection
  - Color of two pixels C1=(R1, G1, B1), C2=(R2, G2, B2)
  - Color Difference

$$D(C1,C2) = \sqrt{(R1-R2)^2+(B1-B2)^2+(G1-G2)^2}$$

  - Compare pixel P and P_{right}, P and P_{bottom}
  - If difference exceeds the threshold K, set P to be white (255, 255, 255); otherwise set P to be black(0, 0, 0)





Advanced DIP functions

- Bit manipulations

```
unsigned char tmp; /*tmp = 'A'=(01000001)2*/
unsigned char bit2;
unsigned char bit1;
unsigned char bit0;
unsigned char andmask[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
unsigned char set1mask[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
unsigned char set0mask[] = {0xfe, 0xfd, 0xfb, 0xf8, 0x8f, 0xbf, 0xdf, 0xef};

bit0 = (tmp & andmask[0]);        /* bit0 = 1 = (01000001)2 & (00000001)2 */
bit1 = (tmp & andmask[1]);        /* bit1 = 0 = (01000001)2 & (00000010)2 */
bit2 = (tmp & andmask[2]);        /* bit2 = 0 = (01000001)2 & (00000100)2 */

/*set bit 0 of tmp to be the same as bit 1 */
if(bit1){
        tmp |= set1mask[0]; /* tmp = (01000001)2 |(00000001)2*/
}
else{
        tmp &= set0mask[0]; /* tmp = (01000001)2 &(11111110)2*/
}
```