ECE 5273
HW 1 Solution

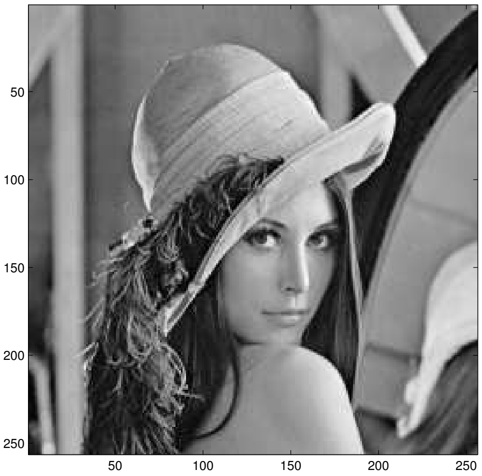
Spring 2025 Dr. Havlicek

Note: This document contains solutions in both Matlab and traditional C.

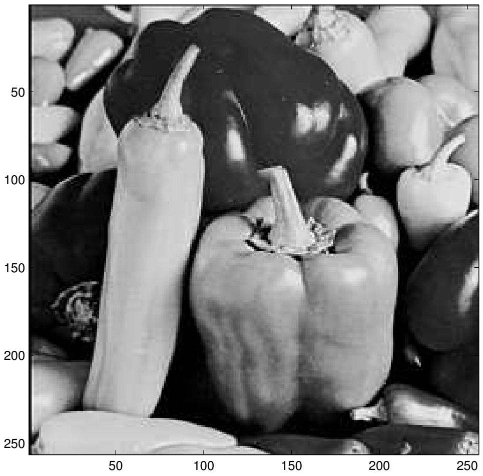
Matlab Solution:

1. Images:

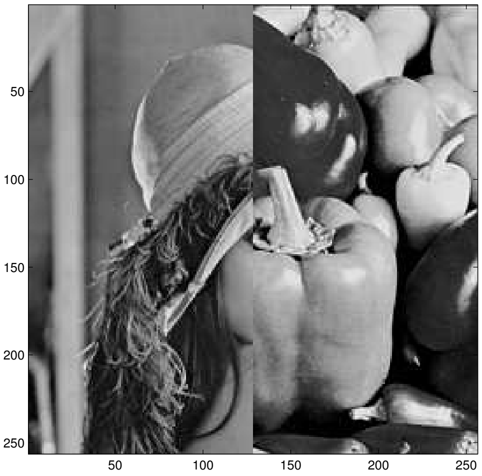
Original Lena Image



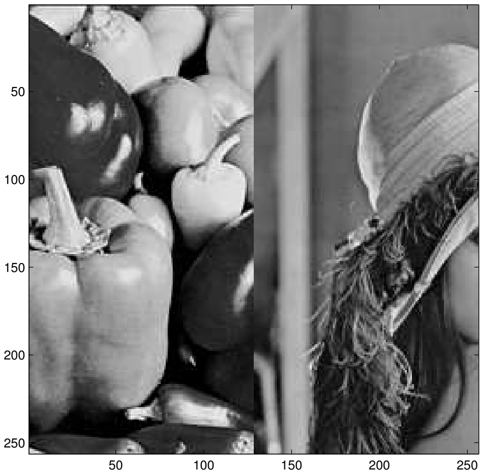
Original Peppers Image



New Image J



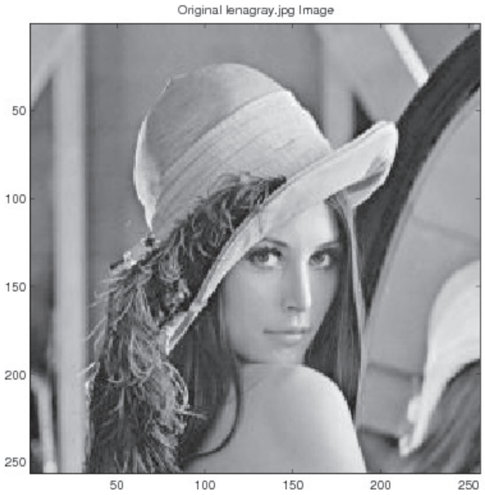
New Image K

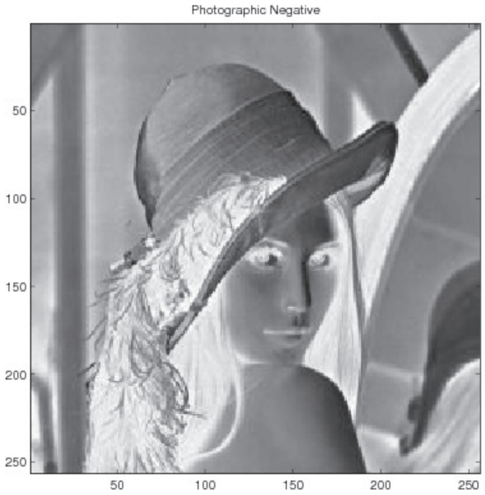


Matlab m-file listing:

```
%
% hw01.m
%
% 02/04/02 jph
% This code is for problem 1.  01/16/2013 jph
%

%
% Open the files containing lena.bin and peppers.bin
%
fidLena = fopen('/home/joebob/Images/lena.bin','r');
fidPeppers = fopen('/home/joebob/Images/peppers.bin','r');

%
% Read the images and transpose
%
[Lena,junk] = fread(fidLena,[256,256],'uchar');
junk
[Peppers,junk] = fread(fidPeppers,[256,256],'uchar');
junk
Lena = Lena'; Peppers = Peppers';

%
% Display each image
%
figure(1);colormap(gray(256));
image(Lena);
axis('image');
title('Original Lena Image');
print -deps M_Lena.eps;

figure(2);colormap(gray(256));
image(Peppers);
axis('image');
title('Original Peppers Image');
print -deps M_Peppers.eps;

%
% Make the image J
%
J = Lena;
J(1:256,129:256) = Peppers(1:256,129:256);
figure(3); colormap(gray(256)); image(J); axis('image');
title('New Image J');
print -deps M_J.eps;

%
% Make the image K
%
K = Lena;                     % this just sets up K to be the right size
K(1:256,1:128) = J(1:256,129:256);
K(1:256,129:256) = J(1:256,1:128);
figure(4); colormap(gray(256)); image(K); axis('image');
title('New Image K');
print -deps M_K.eps;
```

2. Images:

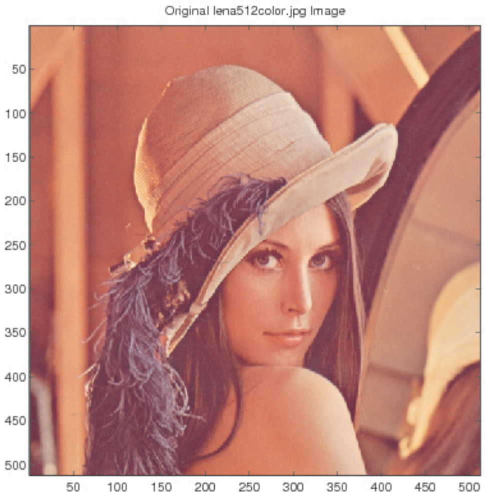


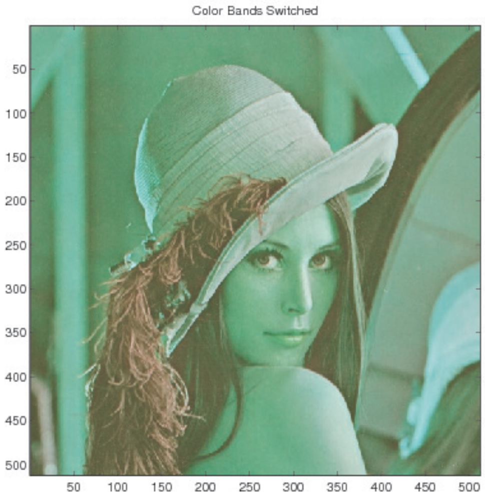


Matlab m-file listing:

```
%
% Read & display lenagray.jpg image
%
J1 = imread('lenagray.jpg','jpg');
figure(1);colormap(gray(256));
image(J1);
axis('image');
title('Original lenagray.jpg Image');
%
% Print as "eps" for report
%
print -deps P2J1.eps;
%
% Make J2 a photographic negative & display
%
J2 = 255 - J1;
figure(2);colormap(gray(256));
image(J2);
axis('image');
title('Photographic Negative');
%
% Write out result as JPEG
%
imwrite(J2,'LenaNegative.jpg','jpg');
%
% Print as "eps" for report
%
print -deps P2J2.eps;
```

3. Images:





Matlab m-file listing:

```
%
% Read & display original color jpg image
%
J1 = imread('lena512color.jpg','jpg');
figure(1);
image(J1);
axis('image');
title('Original lena512color.jpg Image');
%
% Print as color "eps" for report
%
print -depsc P3J1.eps;
%
% Make J2 by swapping the RGB color bands
%
J2 = J1;
J2(:,:,1) = J1(:,:,3);
J2(:,:,2) = J1(:,:,1);
J2(:,:,3) = J1(:,:,2);
%
% Display J2 and write out as JPEG
%
figure(2);
image(J2);
axis('image');
title('Color Bands Switched');
imwrite(J2,'LenaColorSwitch.jpg','jpg');
%
% Print as color "eps" for report
%
print -depsc P3J2.eps;
```

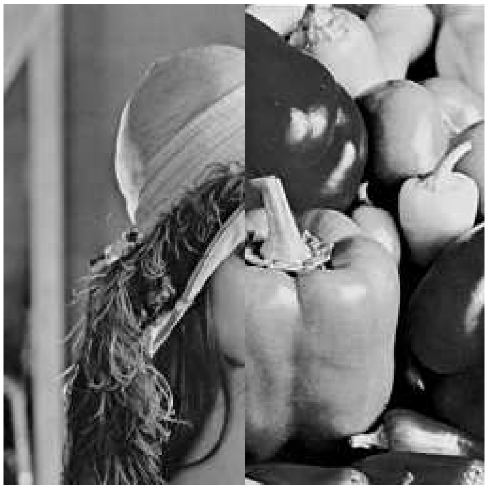
C Solution:

1. Images:

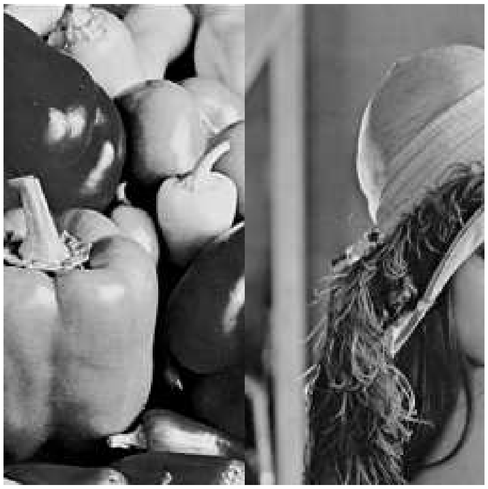
Original Lena Image

Original Peppers Image

New Image **J**



New Image **K**



C program listing:

```
/*
 * hw01.c:
 *
 *   - Read in two size x size BYTE images I1 and I2.
 *   - Make image J so left half is equal to left half of I1 and right half
 *       is equal to right half of I2.  Write J to disk.
 *   - Make image K so it is equal to J with right and left halves swapped.
 *       Write K to disk.
 *
 *   - The input images must be square.
 *
 * 02/04/2002 jph
 *
 *   02/05/2003: histograms of J and K are no longer required. jph.
 *   01/15/2013: this code is for problem 1 only. jph.
 *
 */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>

#define BYTE unsigned char

/*
 * Function Prototypes (forward declarations)
 */
void disk2byte();
void byte2disk();


/*----------------------------------------------------------------------*/
/*   MAIN                                                               */
/*----------------------------------------------------------------------*/

main(argc,argv)

  int     argc;
  char   *argv[];
{

  int      size;                  /* num rows/cols in images */
  int      so2;                   /* size / 2 */
  int      i;                     /* counter */
  int      row;                   /* image row counter */
  int      col;                   /* image col counter */
  BYTE    *I1;                    /* input image I1 */
  BYTE    *I2;                    /* input image I2 */
  BYTE    *J;                     /* output image J */
  BYTE    *K;                     /* output image K */
  char    *InFn1;                 /* input filename for image I1 */
  char    *InFn2;                 /* input filename for image I2 */
  char    *OutFnJ;                /* output filename for image J */
  char    *OutFnK;                /* output filename for image K */

 /*
  * Check for proper invocation, parse args
  */
  if (argc != 6) {
    printf("\n%s: Swap image halves for hw01.",argv[0]);
    printf("\nUsage: %s size InFn1 InFn2 OutFnJ OutFnK\n",
           argv[0]);
    exit(0);
  }
  size = atoi(argv[1]);
  if (size % 2) {
    printf("\n%s: size must be divisible by 2.\n",argv[0]);
    exit(0);
  }
  InFn1 = argv[2];
  InFn2 = argv[3];
  OutFnJ = argv[4];
  OutFnK = argv[5];

  so2 = size >> 1;

 /*
  * Allocate image arrays
  */
  if ((I1 = (BYTE *)malloc(size*size*sizeof(BYTE))) == NULL) {
    printf("\n%s: free store exhausted.\n",argv[0]);
    exit(-1);
  }
  if ((I2 = (BYTE *)malloc(size*size*sizeof(BYTE))) == NULL) {
    printf("\n%s: free store exhausted.\n",argv[0]);
    exit(-1);
  }
  if ((J = (BYTE *)malloc(size*size*sizeof(BYTE))) == NULL) {
    printf("\n%s: free store exhausted.\n",argv[0]);
    exit(-1);
  }
  if ((K = (BYTE *)malloc(size*size*sizeof(BYTE))) == NULL) {
    printf("\n%s: free store exhausted.\n",argv[0]);
    exit(-1);
  }

 /*
  * Read input images
  */
  disk2byte(I1,size,size,InFn1);
  disk2byte(I2,size,size,InFn2);

 /*
  * Make output image J: left half is I1, right half is I2
  */
  for (i=row=0; row < size; row++) {
    for (col=0; col < so2; col++,i++) {
      J[i] = I1[i];
    }
    for ( ; col < size; col++,i++) {
      J[i] = I2[i];
    }
  }

 /*
  * Make output image K: swap left and right halves of J
  */
  for (row=0; row < size; row++) {
    for (col=0; col < so2; col++) {
      K[row*size + col] = J[row*size + col+so2];
    }
    for ( ; col < size; col++) {
      K[row*size + col] = J[row*size + col-so2];
    }
  }

 /*
  * Write the output images
  */
  byte2disk(J,size,size,OutFnJ);
  byte2disk(K,size,size,OutFnK);
```

```
  return;
} /*----------------  Main  ----------------------------------------------*/


/*----------------------------------------------------------------------
 * disk2byte.c
 *
 *   function reads an unsigned char (byte) image from disk
 *
 *
 *  jph 15 June 1992
 *
 -----------------------------------------------------------------------*/

void disk2byte(x,row_dim,col_dim,fn)

  BYTE    *x;             /* image to be read */
  int     row_dim;        /* row dimension of x */
  int     col_dim;        /* col dimension of x */
  char   *fn;             /* filename */
{

  int fd;               /* file descriptor */
  int n_bytes;          /* number of bytes to read */

 /*
  * detect zero dimension input
  */
  if ((row_dim==0) || (col_dim==0)) return;

 /*
  * create and open the file
  */
  if ((fd = open(fn, O_RDONLY))==-1) {
    printf("\ndisk2byte.c : could not open %s !",fn);
    return;
  }

 /*
  * read image data from the file
  */
  n_bytes = row_dim * col_dim * sizeof(unsigned char);
  if (read(fd,x,n_bytes) != n_bytes) {
    printf("\ndisk2byte.c : complete read of %s did not succeed.",fn);
  }

 /*
  * close file and return
  */
  if (close(fd) == -1) printf("\ndisk2byte.c : error closing %s.",fn);
  return;
}



/*----------------------------------------------------------------------
 * byte2disk.c
 *
 *   function writes an unsigned char (byte) image to disk
 *
 *
 *  jph 15 June 1992
 *
 -----------------------------------------------------------------------*/

void byte2disk(x,row_dim,col_dim,fn)

  BYTE  *x;              /* image to be written */
  int   row_dim;         /* row dimension of x */
  int   col_dim;         /* col dimension of x */
  char *fn;              /* filename */
{

  int fd;               /* file descriptor */
  int n_bytes;          /* number of bytes to read */

 /*
  * detect zero dimension input
  */
  if ((row_dim==0) || (col_dim==0)) return;

 /*
  * create and open the file
  */
  if ((fd = open(fn, O_WRONLY | O_CREAT | O_TRUNC, 0644))==-1) {
    printf("\nbyte2disk.c : could not open %s !",fn);
    return;
  }

 /*
  * write image data to the file
  */
  n_bytes = row_dim * col_dim * sizeof(unsigned char);
  if (write(fd,x,n_bytes) != n_bytes) {
    printf("\nbyte2disk.c : complete write of %s did not succeed.",fn);
  }

 /*
  * close file and return
  */
  if (close(fd) == -1) printf("\nbyte2disk.c : error closing %s.",fn);
  return;
}
```

2. See the Matlab solution: for HW1, it's too much trouble to read JPEG files in C.
3. See the Matlab solution.