Problem 1.
Write a program to determine the number of foreground object regions in a binary image. Label the pixels according to the region and, for each object region, compute:

1. The area.
2. The centroid.
3. The orientation of the major axis using second moments.

The output should be a set of images, one per region. Each of these images should contain:

1. The location of the centroid marked somehow (e.g., a bright point).
2. The orientation indicated somehow (e.g., as an arrow or line segment).

Test your program with binary1.bmp (257x22) and binary2.bmp (247x261). In these images, black denotes the background and white denotes the foreground object.

Problem 2.
Write another program to classify each pixel in the image with a color label. For the image in face.bmp, label the black pixels (background) as 0, the red pixels as 1 and the green pixels as 2. Show this output image.

Test your program on the face.bmp.

Extra credit (up to 20%): .
Extend the second program (if necessary) to handle the real photo (mixed_fruit.bmp) to label pixels corresponding to the banana as 1, the oranges as 2, and the apple as 3, and everything else as 0. Output the label image. Then using our algorithm from part 1, count the number of bananas, apples, and oranges.

What to hand in: .
1. Source code. Try to write your program in a clear, readable style.

2. Output images. The output images are (binary) images that contain segmentation for each object in the input image. Conventionally, we use black for background and white for object.

3. For problem 1, your output should be a set of 4 images for each given input. Also, give a text file describes the area, the centroid and the orientation of each region.

4. For problem 2, produce a 2 (black and white) images, each of them represents the red and green region separately.

5. For extra problem, produce 3 images as above, each for apples, oranges and bananas.

6. Zip all your files and send to d1vu@cs.ucsd.edu.

All the test images can be found at http://www-cse.ucsd.edu/classes/sp03/cse190-b/hw1/

In Matlab, you can read image with im = imread('binary1.bmp','bmp').

Note that when programming in matlab, you can only use functions in the image processing toolbox (or related toolboxes) for I/O and display, not for the core assignment.

For those using C/C++, you can use the bmp_io routine (found at http://www-cse.ucsd.edu/classes/sp03/cse190-b/hw1/bmp_io.tar). Two functions you may want to use:

```c
int bmp_read ( char *filein_name, 
               int *xsize, int *ysize, int *bsize, 
               unsigned char **rarray, 
               unsigned char **garray, 
               unsigned char **barray )
```

```c
int bmp_write ( char *filein_name, 
                int xsize, int ysize, int bsize, 
                unsigned char *rarray, 
                unsigned char *garray, 
                unsigned char *barray )
```

See main.c for the example of using these functions.