Assignment1: (Due at 7 pm on Sept 22, 1998)

Goal? The goal of this assignment is to familiarize you with manipulating arrays. You will have to write Ada code to read, write and manipulate images.

Coding: Your code writing task can be divided into two parts:

1. You have to write a package to implement the image abstract data type using arrays. The ADT should conform to the specifications in at the back. The Image ADT should allow you
   (a) to read images from a file and to save images to a file.
   (b) to set and get sizes of images.
   (c) to set and get pixel values of an image.
   (d) to indicate whether a pixel is inside an image or not.
2. Write a driver program which interacts with the above Image ADT and the user. The user should have options to read in an image file, threshold the image based on an user specified number, and save the thresholded image to a file. The thresholding options should be the following two:
   (a) to threshold a image based on an **single** threshold, N. The thresholding operation is really simple. Given a gray level image, I, you have to create an output image, O, such that for each pixel location (i, j):

$$O(i,j) = \begin{cases} 255 & \text{if } I(i,j) > N \\ 0 & \text{if } I(i,j) \leq N \end{cases}$$

   (b) to threshold a image based on **double** thresholds, N_1 and N_2. Given a gray level image, I, you have to create an output image, O, such that for each pixel location (i, j):

$$O(i,j) = \begin{cases} 255 & \text{if } I(i,j) > N_1 \text{ and } I(i,j) <= N_2 \\ 0 & \text{otherwise} \end{cases}$$

```
with text_io;

package Image_Data_Type is

   -- This package implements an ADT to represent images

   -- Structure:  The image is represented as an array of pixel values

OUT_OF_BOUNDS : Exception;
    -- Raised if an attempt is made to access
    -- image pixels outside the image
NOT_PGM: exception;
    -- Raised if the image file to be read is not in ascii PGM format

    type Image_Array is private;
```

```
    ------------------------------------------------------------------------
    procedure Read (File : in  Text_IO.File_Type;
    Image: out Image_Array);
    -- Function:        Reads in the Image from File
    -- Preconditions:   None
    -- Postconditions:  All the image pixels are read in and the image
    --                  width and height recorded in the appropriate fields.
    ------------------------------------------------------------------------
    procedure Save (File : in  Text_IO.File_Type;
    Image: in Image_Array);
    -- Function:        Write out the Image to the File in PGM format
    -- Preconditions:   None
    -- Postconditions:  The written image file is in PGM format

    ---------------------------------------------------------------------------
    procedure Get_Size (Image:in Image_Array;
Row_Num: out positive;
Col_Num: out positive);

    -- Function:   Returns the total number of Rows and Columns in the Image
    -- Preconditions:   None
    -- Postconditions: The values are returned through the procedure parameters

    ---------------------------------------------------------------------------
    procedure Set_Size (Image: out Image_Array;
Row_Num: in positive;
Col_Num: in positive);

    -- Function: Sets the total number of Rows and Columns in the Image
    --           if Image is null then it creates a new record
    -- Preconditions:   None
    -- Postconditions: The values are returned through the Image parameter

    ---------------------------------------------------------------------------
    function Get_Pixel (Image:in Image_Array;
Row: positive;
Col: positive) return integer;

    -- Function:   Returns the pixel value at Image(Row, Col)
    -- Preconditions:   None
    -- Postconditions:  Should not change the Image
    -- Exception:  OUT_OF_BOUNDS exception raised if (Row, Col) falls
    --              outside the image
    ---------------------------------------------------------------------------
    procedure Set_Pixel (Image:in out Image_Array;
 Row: positive;
 Col: positive;
 Value: integer);
```

```
    -- Function:   Sets the pixel value at Image(Row, Col) to Value
    -- Preconditions:   None
    -- Postconditions:  Should change the Image at (Row, Col)
    -- Exception:  OUT_OF_BOUNDS exception raised if (Row, Col) falls
    --              outside the image
    ------------------------------------------------------------------------
    function Inbounds (Image: in Image_Array;
       X: Integer;
       Y: Integer) return Boolean;
    -- Function: Returns True if the pixel (X,Y) is inside the Image
    -- Preconditions: Image should be initialized
    -- Postconditions: None
    -- Exception:  OUT_OF_BOUNDS exception raised if (X, Y) falls
    --              outside the image
    ------------------------------------------------------------------------

private

   type Image_array_type is array(1..250,1..250) of integer;

   type Image_Array_Record is
      record
     Rows: positive; -- stores the total number of rows in the image
     Cols: positive; -- stores the total number of cols in the image
     Image: Image_array_type;
   end record;

   type Image_Array is access Image_Array_Record;
end Image_Data_Type;
```