

Mark Skidmore/Guilherme Martins

CS302 Data Structures
Fall 2008 - Dr. George Bebis
Programming Assignment 1
Due Date: 9/25/2008

I. Introduction

The purpose of this program is to read, manipulate and write digital images. This assignment provides an introduction to some simple image processing algorithms which are described in Section III. In addition to providing a review of writing class and client functions, the algorithms also review the usage of array manipulation, constructors, destructors, and overloading C++ operators.

The program consists of one class, image.h/cpp, the driver program, processImageMain.cpp, and the following supporting files:

- ReadImage.cpp (supplied by Dr. Bebis)
- ReadImageHeader.cpp (supplied by Dr. Bebis)
- WriteImage.cpp (supplied by Dr. Bebis)
- processImage.make (make file)
- processImage.bin (executable)
- object files

The program defines and uses the ImageType object throughout the program to read, manipulate and write images. An on-screen menu is displayed which allows the user to perform simple image manipulation functions as described in Section II.

II. Program Usage

The program is simple to use. In a command window, navigate to the folder containing the program files. Then from the command line, simply run the executable by typing ./processImage.bin and follow the on-screen menu as shown below in Illustration 1:

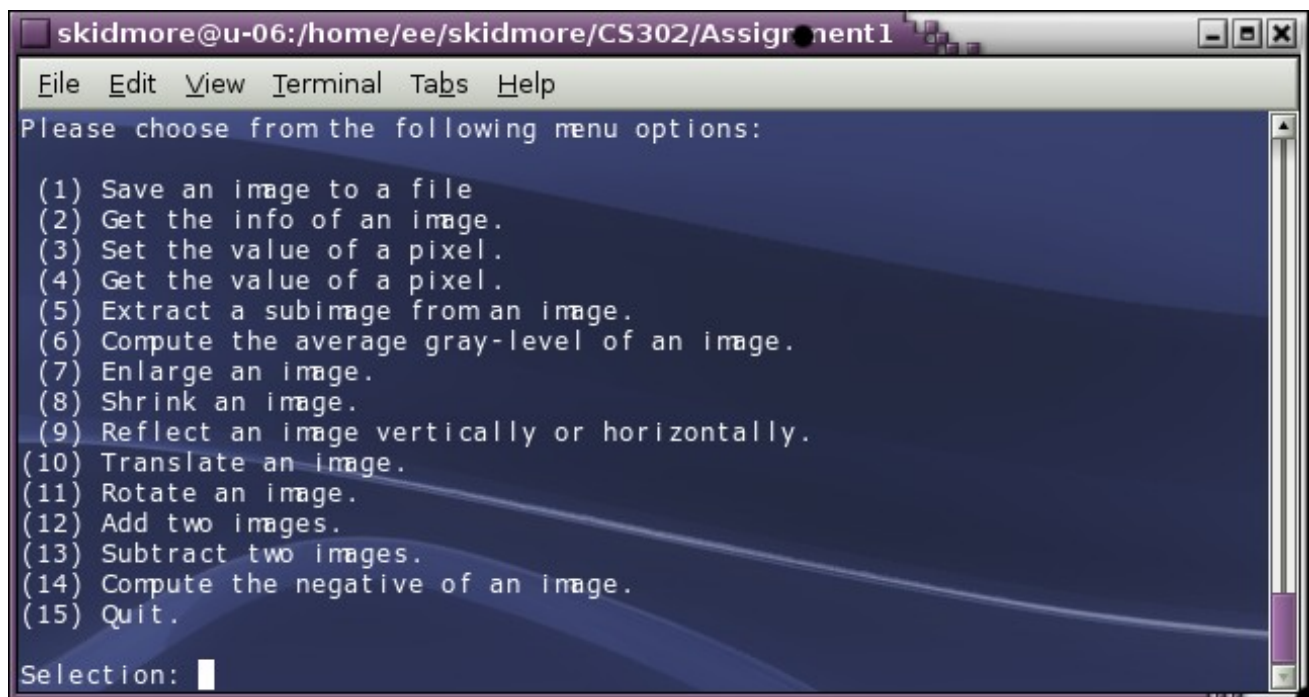


Illustration 1: Menu List

Execution of each menu item is demonstrated below.

1) Save an Image. This shows the user how the image will be written to a file. (This is a function call from Professor Bebis' writeimage.cpp). Illustration 2 shows an original file while Illustration 3 shows the same file read and written to disk.

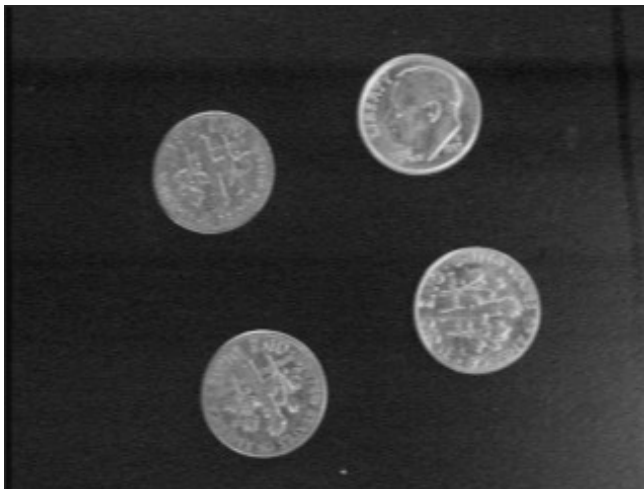


Illustration 2: Dimes.pgm

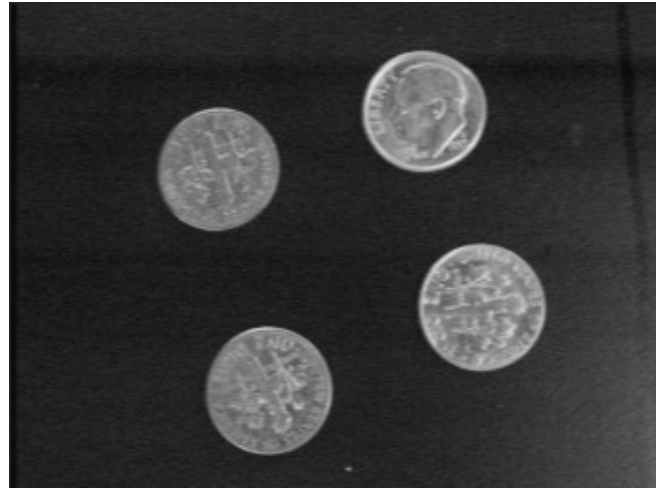


Illustration 3: Output.pgm

2) Get image Info. Implemented by Professor Bebis', this function retrieves information about a given image file. Illustration 4 shows the results of the user selecting menu item 2 and the results.

```
▼ martins@u-07:/home/other/martins/Desktop/Assignment1
File Edit View Terminal Tabs Help
(13) Subtract two images.
(14) Compute the negative of an image.
(15) Quit.

Selection: 2

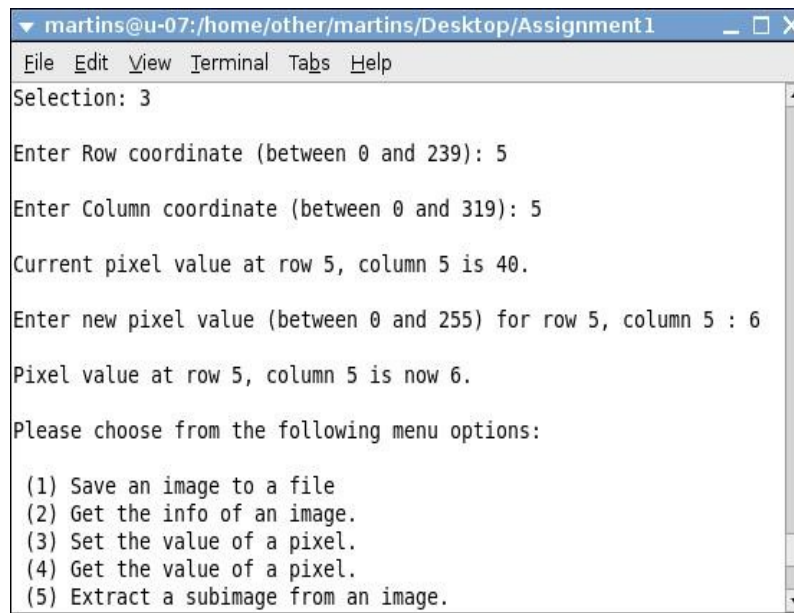
The image contains 240 rows, 320 columns and the maximum pixel
is 255.

Please choose from the following menu options:

(1) Save an image to a file
(2) Get the info of an image.
(3) Set the value of a pixel.
```

Illustration 4: Get info of an image

3) Set the value of a pixel. This function was provided by Dr. Bebis. Shown in Illustration 5, the user inputs the row and column location and new pixel value.



```
▼ martins@u-07:/home/other/martins/Desktop/Assignment1 _ □ X
File Edit View Terminal Tabs Help
Selection: 3

Enter Row coordinate (between 0 and 239): 5
Enter Column coordinate (between 0 and 319): 5
Current pixel value at row 5, column 5 is 40.

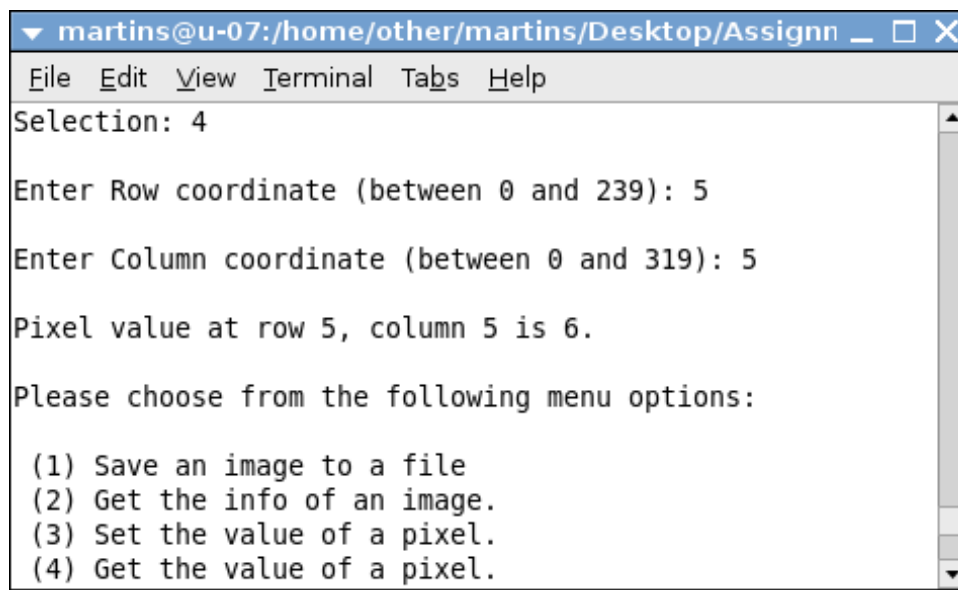
Enter new pixel value (between 0 and 255) for row 5, column 5 : 6
Pixel value at row 5, column 5 is now 6.

Please choose from the following menu options:

(1) Save an image to a file
(2) Get the info of an image.
(3) Set the value of a pixel.
(4) Get the value of a pixel.
(5) Extract a subimage from an image.
```

Illustration 5: Set the value of a pixel

4) Get the value of a pixel, this function was provided by Dr. Bebis. Illustration 6 shows the user being prompted for the row and column location of the pixel value he/she wishes to retrieve.



```
▼ martins@u-07:/home/other/martins/Desktop/Assignnn _ □ X
File Edit View Terminal Tabs Help
Selection: 4

Enter Row coordinate (between 0 and 239): 5
Enter Column coordinate (between 0 and 319): 5
Pixel value at row 5, column 5 is 6.

Please choose from the following menu options:

(1) Save an image to a file
(2) Get the info of an image.
(3) Set the value of a pixel.
(4) Get the value of a pixel.
```

Illustration 6: Get the pixel value

5) Extract a sub-image from an image. This function has user-entered upper-right and left corner, row and column coordinate. The program then extracts a sub-image from the original image. The user input is shown in Illustration 7 along with the output below in Illustration 8 below.

```
File Edit View Terminal Tabs Help
(15) Quit.

Selection: 5
Enter the Upper Left corner extraction coordinates:
Upper-Left Row Coordinate: 60
Upper-Left Column Coordinate: 80
Now enter the Lower Right corner extraction coordinates:
Lower-Right Row Coordinate: 120
Lower-Right Column Coordinate: 240

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Destructor...
Departing Destructor...

Enter the name of your extracted output file:
dimes.sub.pgm

Please choose from the following menu options:

(1) Save an image to a file
Illustration 7: Menu Item 5
```

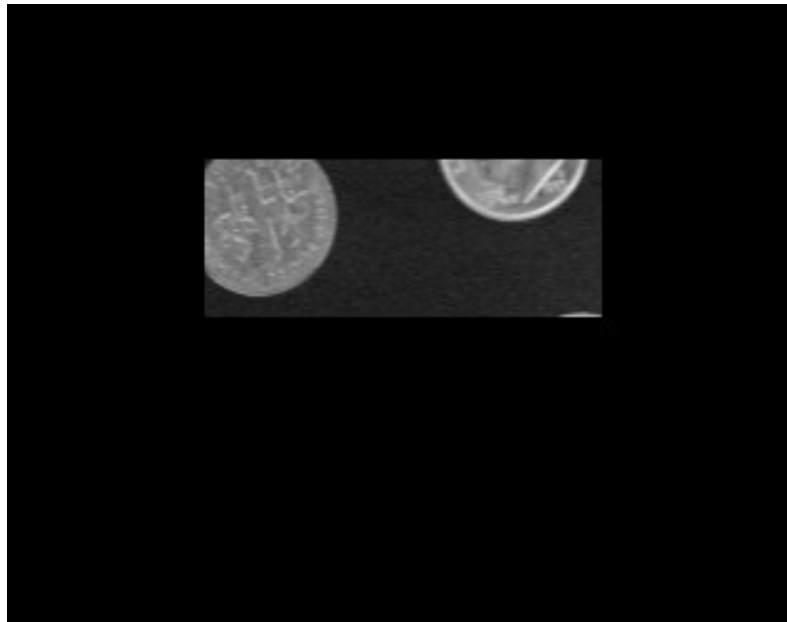
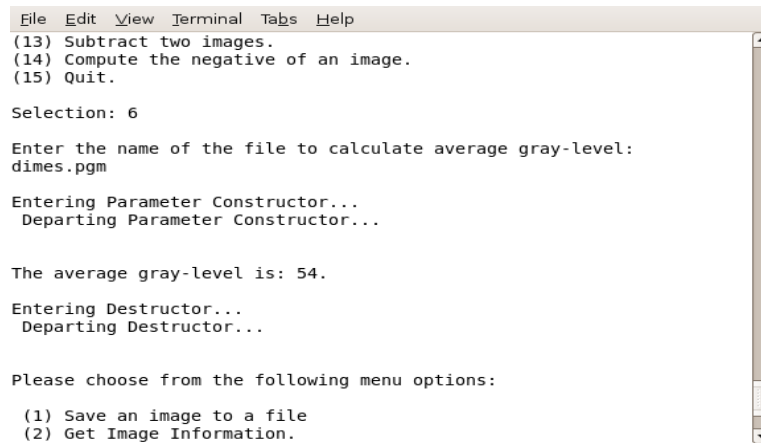


Illustration 8: Extracted Image

6) Calculate the Average Gray Scale, This function will take a given image, walk through its array and calculate the average of the gray scale values in the image. The user specifies an image and the output is displayed to the screen as shown in Illustration 9 below.



```
File Edit View Terminal Tabs Help
(13) Subtract two images.
(14) Compute the negative of an image.
(15) Quit.

Selection: 6

Enter the name of the file to calculate average gray-level:
dimes.pgm

Entering Parameter Constructor...
Departing Parameter Constructor...

The average gray-level is: 54.

Entering Destructor...
Departing Destructor...

Please choose from the following menu options:
(1) Save an image to a file
(2) Get Image Information.
```

Illustration 9: Average Gray Level

7) Enlarge an Image. This function enlarges an image by user defined “factor”. The basic algorithm was provided by Dr. Bebis but still needed to be implemented by walking through and multiplying the image array by the factor. The selection of the factor is shown in Illustration 10 and the resulting image is shown in Illustration 11.

```

File Edit View Terminal Tabs Help
(15) Quit.

Selection: 7

Entering Default Constructor...
Departing Default Constructor...

Enter the Enlarge Factor Number: 2

Image being Enlarged...

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Destructor...
Departing Destructor...

Entering Overloaded = Operator...
Departing Overloaded = Operator...

Entering Destructor...
Departing Destructor...

Enter the name of your enlarged output file:
dimesbig.pgm

Entering Destructor...
Departing Destructor...

Please choose from the following menu options:

(1) Save an image to a file

```

Illustration 10: Enlarge Selection

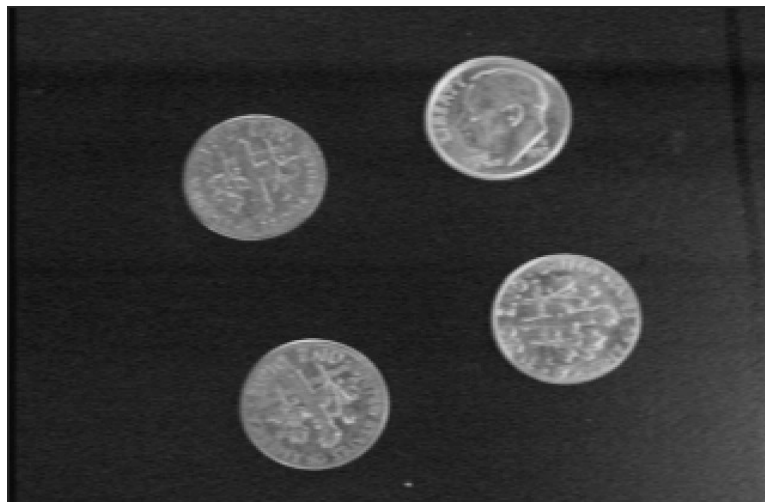


Illustration 11: Enlarged Image

8) Shrink an Image. This function reduces an image by a user defined “factor”. The basic algorithm was provided by Dr. Bebis but still needed to be implemented by walking through and dividing the image array by the factor. The selection of the factor is shown in Illustration 12 and the resulting image is shown in Illustration 13.

```
File Edit View Terminal Tabs Help
(15) Quit.

Selection: 8

Entering Default Constructor...
Departing Default Constructor...

Enter the Shrink Factor Number: 2

Image being shrunk...

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Destructor...
Departing Destructor...

Entering Overloaded = Operator...
Departing Overloaded = Operator...

Entering Destructor...
Departing Destructor...

Enter the name of your shrunk output file:
dimesmall.pgm

Entering Destructor...
Departing Destructor...

Please choose from the following menu options:

(1) Save an image to a file
```

Illustration 12: Shrink Selection

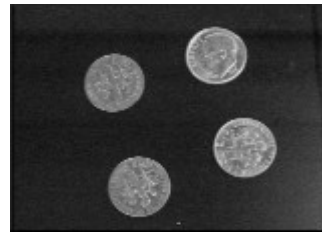


Illustration 13: Shrunk Image

9) Reflect an Image. This function will be split into two parts, one for the reflection vertically and one for horizontally, this does as the name says a reflection upon the X and Y axis. In Illustrations 14 and 15, notice the user input for vertical and the corresponding output.

```
File Edit View Terminal Tabs Help
(13) Subtract two images.
(14) Compute the negative of an image.
(15) Quit.

Selection: 9
Image being reflected...

Entering Parameter Constructor...
Departing Parameter Constructor...

Would you like to reflect your image (V)ertically (about the X-axis) or (H)orizontally (about the Y-axis)?v

Entering Destructor...
Departing Destructor...

Enter the name of your reflected output file:
dimesv.pgm

Please choose from the following menu options:

(1) Save an image to a file
(2) Get Image Information.
```

Illustration 14: Reflect Vertically



Illustration 15: Dimes Vertical

Illustrations 16 and 17 show horizontal selection and output, respectively.

```
File Edit View Terminal Tabs Help
(13) Subtract two images.
(14) Compute the negative of an image.
(15) Quit.

Selection: 9
Image being reflected...

Entering Parameter Constructor...
Departing Parameter Constructor...

Would you like to reflect your image (V)ertically (about the X-axis) or (H)orizontally (about the Y-axis)?H

Entering Destructor...
Departing Destructor...

Enter the name of your reflected output file:
dimesH.pgm

Please choose from the following menu options:

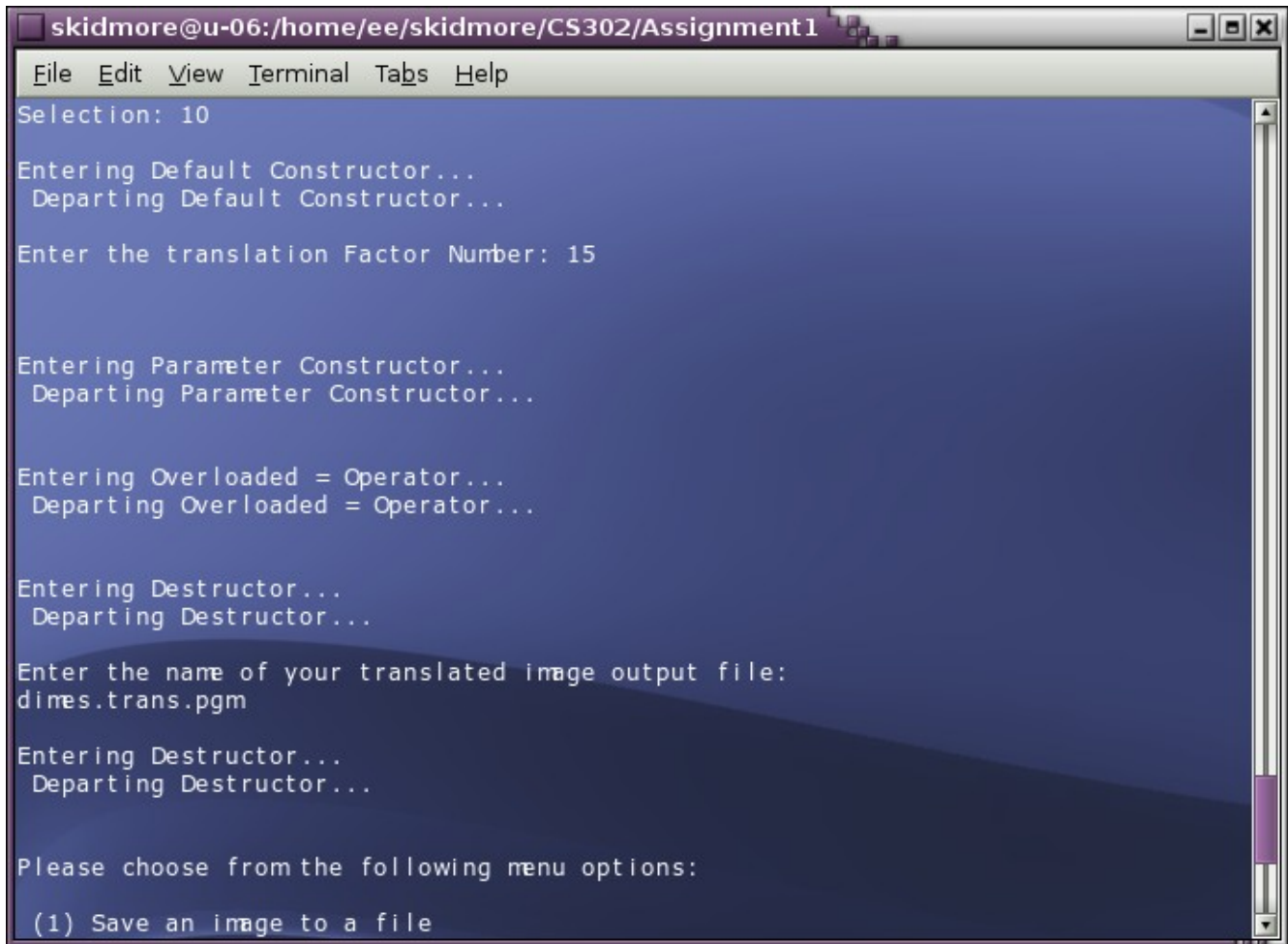
(1) Save an image to a file
(2) Get Image Information.
```

Illustration 16: Reflect Horizontally



Illustration 17: Horizontal Output

10) Translate an image. Shifts an image to its right and down. User is asked to enter the amount (factor) by which he/she would like the image shifted and the name of the output file as shown below in Illustration 18.



```
skidmore@u-06:/home/ee/skidmore/CS302/Assignment1
File Edit View Terminal Tabs Help
Selection: 10
Entering Default Constructor...
Departing Default Constructor...
Enter the translation Factor Number: 15
Entering Parameter Constructor...
Departing Parameter Constructor...
Entering Overloaded = Operator...
Departing Overloaded = Operator...
Entering Destructor...
Departing Destructor...
Enter the name of your translated image output file:
dimes.trans.pgm
Entering Destructor...
Departing Destructor...
Please choose from the following menu options:
(1) Save an image to a file
```

Illustration 18: Menu Item 10

Illustration 19 shows the result of translating dimes.pgm by a factor of 15 and writing the translated image to dimes.trans.pgm (Illustration 20).

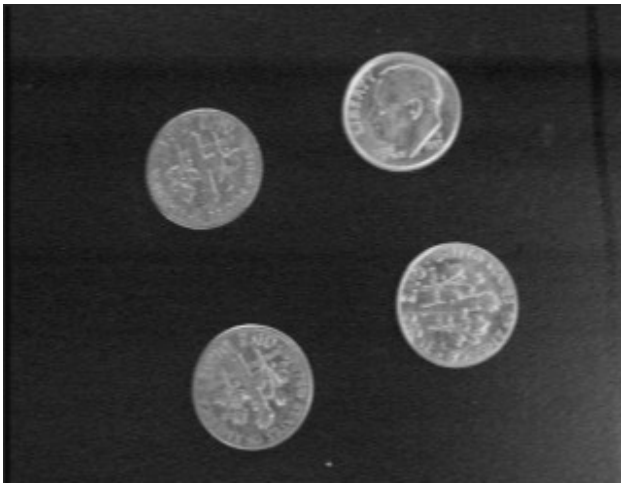


Illustration 19: dimes.pgm

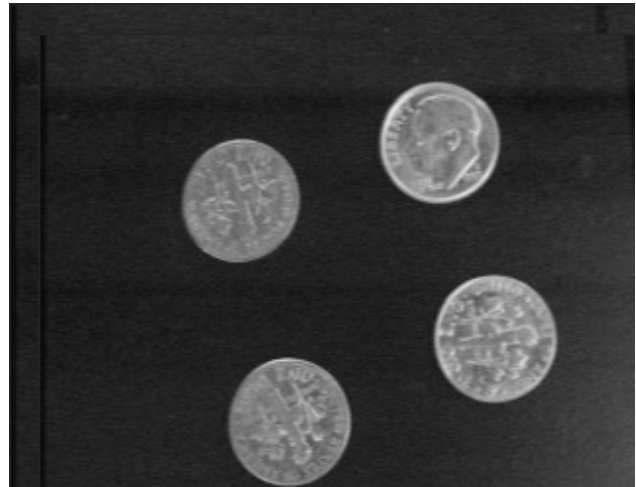


Illustration 20: Translated dimes

11) Rotate an image. Rotates an image by a user-specified number of degrees (positive rotates counter-clockwise, negate rotates clockwise) as shown in illustration 21.

```

skidmore@u-03:/home/ee/skidmore/CS302/Assignment1
File Edit View Terminal Tabs Help
(13) Subtract two images.
(14) Compute the negative of an image.
(15) Quit.

Selection: 11
Image will be rotated about its center coordinates (120, 160).

Enter angle in degrees between -360 and 360 to rotate image: 44
Image being rotated...

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Destructor...
Departing Destructor...

Enter the name of your rotated output file:
dimes.44.pgm

Please choose from the following menu options:

(1) Save an image to a file
(2) Get the info of an image.

```

Illustration 21: Menu Item 11

Illustration 22 below shows a comparison of the original image and a rotated image in Illustration 23.

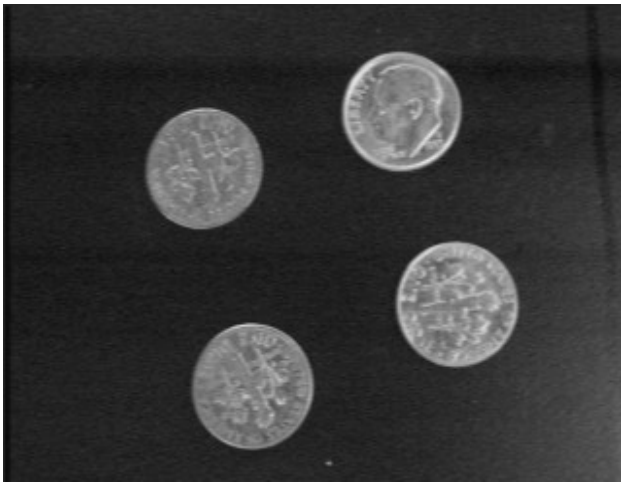


Illustration 22: dimes.pgm

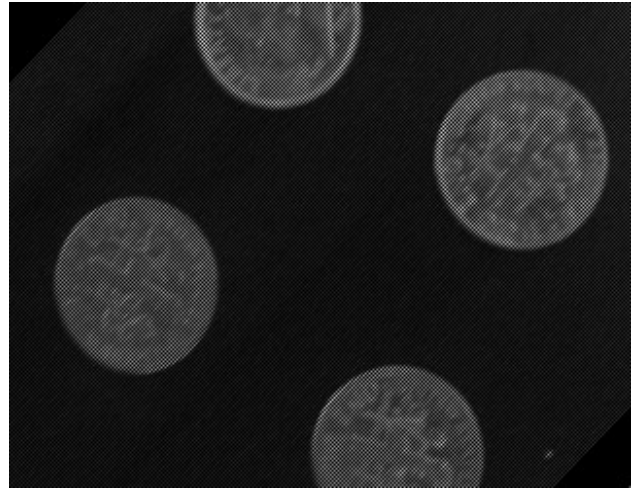
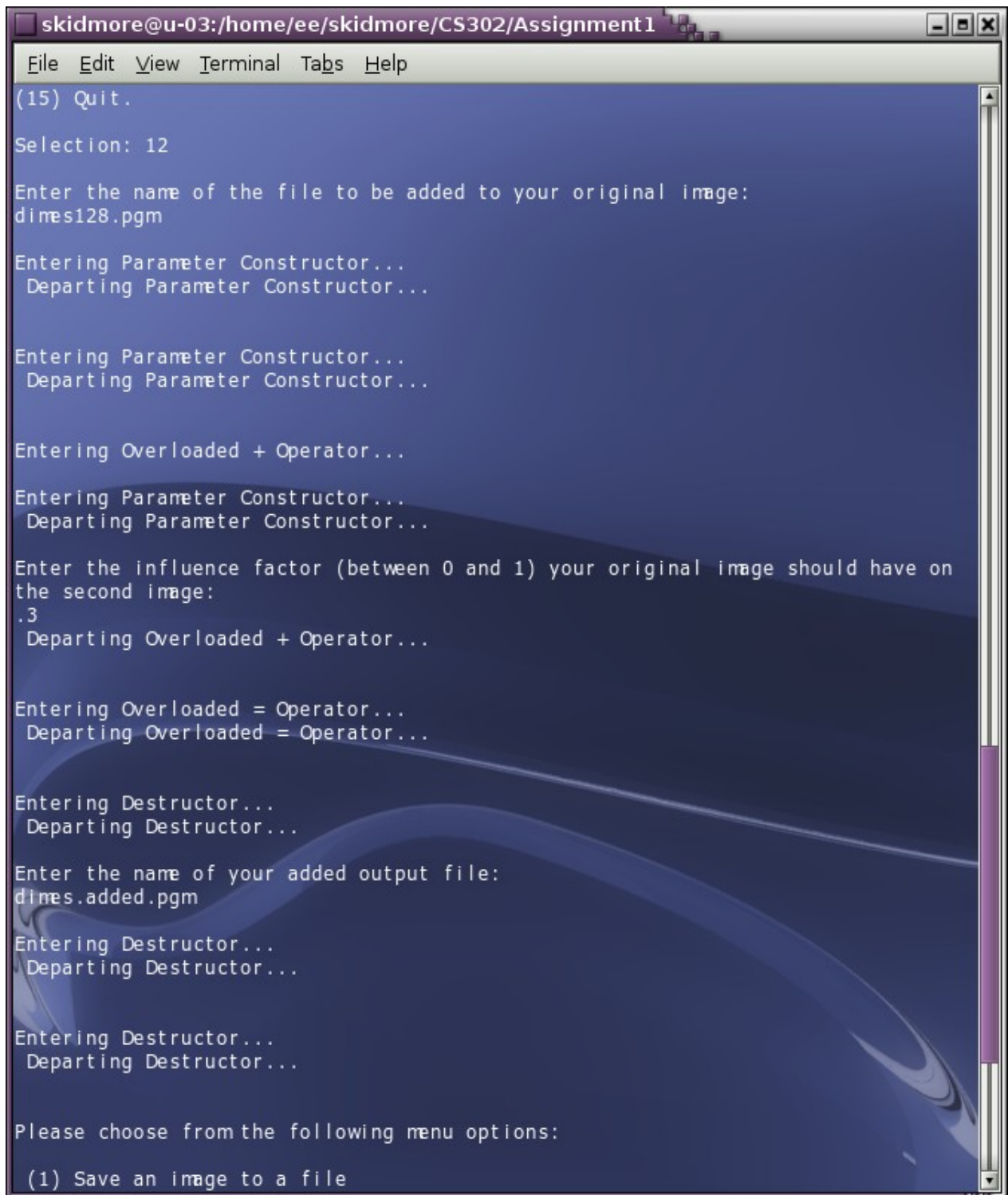


Illustration 23: Rotated dimes

12) Add two images. The user is asked for the file name of a second image to be added to the image already in memory and an influence factor the first image should have on the second. This function uses the overloaded + and = operators to add the two images. Illustration 24 below shows the user menu input.



```
skidmore@u-03:/home/ee/skidmore/CS302/Assignment1
File Edit View Terminal Tabs Help
(15) Quit.
Selection: 12
Enter the name of the file to be added to your original image:
dines128.pgm
Entering Parameter Constructor...
Departing Parameter Constructor...
Entering Parameter Constructor...
Departing Parameter Constructor...
Entering Overloaded + Operator...
Entering Parameter Constructor...
Departing Parameter Constructor...
Enter the influence factor (between 0 and 1) your original image should have on
the second image:
.3
Departing Overloaded + Operator...
Entering Overloaded = Operator...
Departing Overloaded = Operator...
Entering Destructor...
Departing Destructor...
Enter the name of your added output file:
dines.added.pgm
Entering Destructor...
Departing Destructor...
Entering Destructor...
Departing Destructor...
Please choose from the following menu options:
(1) Save an image to a file
```

Illustration 24: Menu Item 12

Illustration 25 shows the original image; Illustration 26 is the image to be added to the original image; Illustration 27 shows the resulting image.

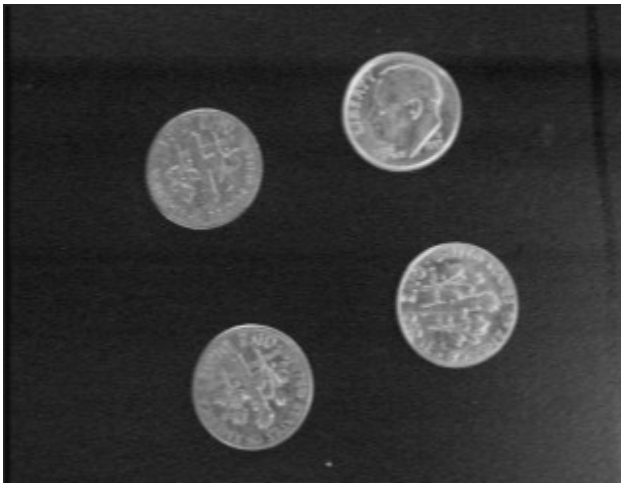


Illustration 25: dimes.pgm

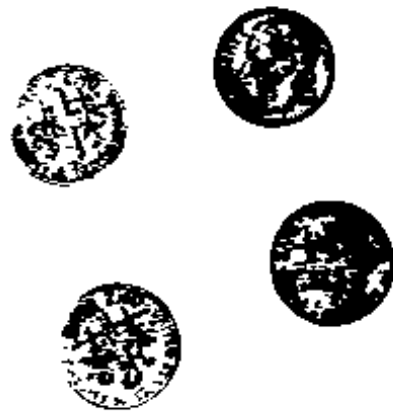
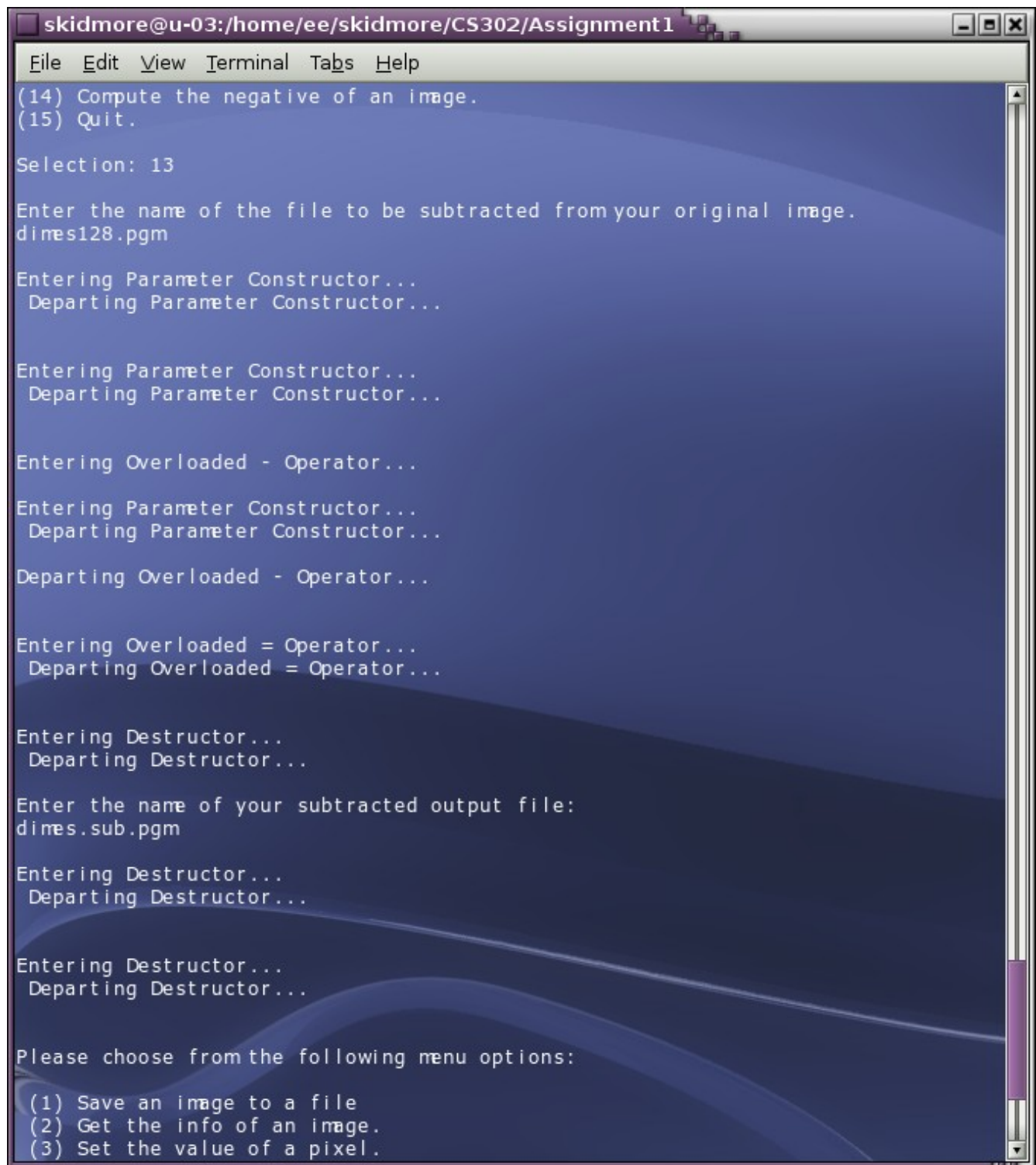


Illustration 26: Additional Image



Illustration 27: Addition of Images

13) Subtracts one image from another. The user is asked for the file name of a second image to be subtracted from the image already in memory. This function uses the overloaded - and = operators to subtract the two images. Illustration 28 below shows the user menu input.



```
skidmore@u-03:/home/ee/skidmore/CS302/Assignment1
File Edit View Terminal Tabs Help
(14) Compute the negative of an image.
(15) Quit.

Selection: 13

Enter the name of the file to be subtracted from your original image.
dimes128.pgm

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Parameter Constructor...
Departing Parameter Constructor...

Entering Overloaded - Operator...

Entering Parameter Constructor...
Departing Parameter Constructor...

Departing Overloaded - Operator...

Entering Overloaded = Operator...
Departing Overloaded = Operator...

Entering Destructor...
Departing Destructor...

Enter the name of your subtracted output file:
dimes.sub.pgm

Entering Destructor...
Departing Destructor...

Entering Destructor...
Departing Destructor...

Please choose from the following menu options:
(1) Save an image to a file
(2) Get the info of an image.
(3) Set the value of a pixel.
```

Illustration 28: Menu Item 13

Illustration 29 shows the original image; Illustration 30 is the image to be subtracted from the original image; Illustration 31 shows the resulting image.

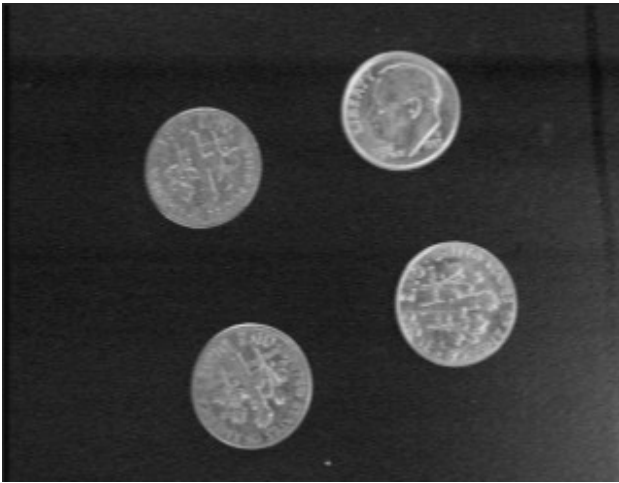


Illustration 29: dimes.pgm

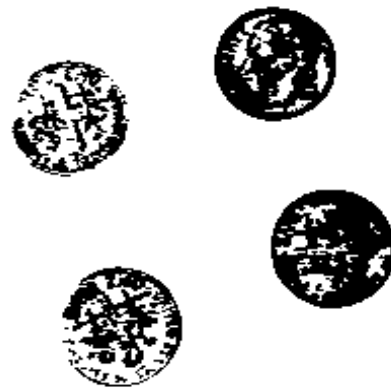


Illustration 30: image128.pgm

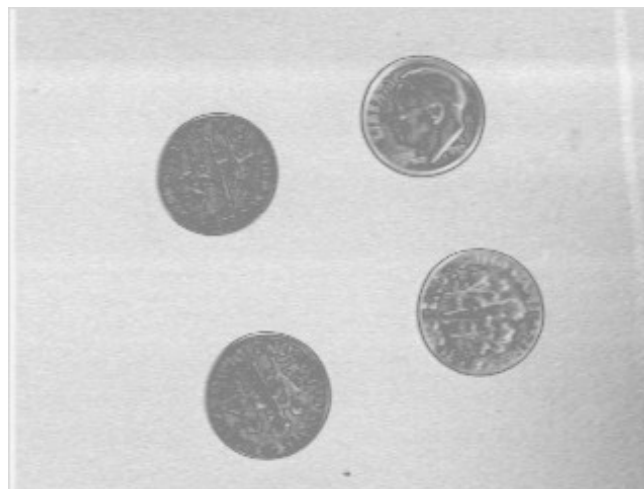


Illustration 31: Difference of Images

14) Calculates the negative of an image. The program prompts the user for an image to negate and the name of an output file as shown in Illustration 32 below.

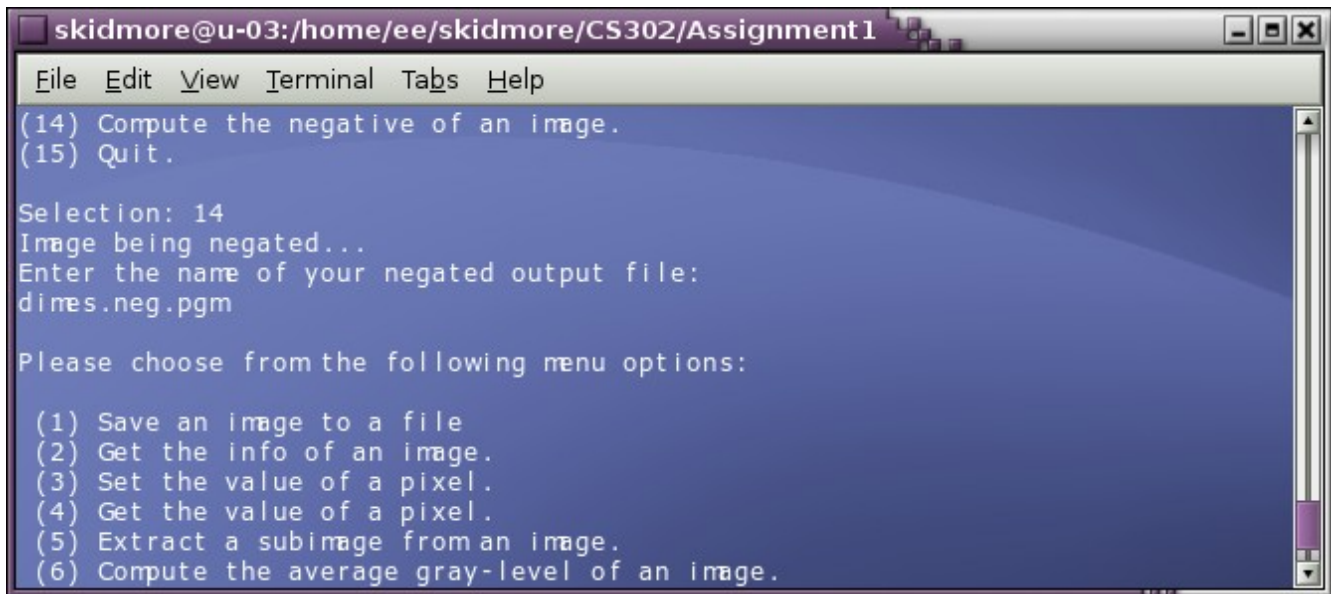


Illustration 32: Menu Item 14

Illustration 33 below shows the original image and its negative.



Illustration 33: dimes.pgm



Illustration 34: Negative Image

III. Program Implementation

A. The Client Functions.

The main driver program, processImageMain.cpp, contains the following functions which are briefly described below: (Note: reference to class methods in this section are abbreviated by omitting parameters; full method descriptions are provided in Section III.B.)

void executeCases(int choice, int N, int M, int Q, ImageType& image);

Provides the switch statement to accept and execute user's menu choice. Also accepts row (N), column (M), number of gray-level values and an image object which are parameters that are used throughout the switch.

void setPixelValues(int N, int M, ImageType& image);

Prompts user for desired row, column and pixel value parameters in order to set a user-defined pixel value at a user-defined row and column coordinate. Receives row (N), column (M) and image object parameters from executeCases() function that are used to validate user input. Calls class method setPixelVal(row, column, PixelValue). Prior and post- pixel coordinates and values are displayed on the screen.

void getPixelValues(int N, int M, ImageType& image);

Prompts user for desired row, column and pixel value parameters in order to get a pixel value at a user-defined row and column coordinate. Accepts row (N), column (M) and image object parameters from executeCases() function that are used to validate user input. Calls class method getPixelVal(row, column, PixelValue). Pixel coordinates and values are displayed on the screen.

void extractImage(int N, int M, ImageType& image);

Prompts user for desired upper-left and lower-right, row and column corner coordinates which are used in computing the boundaries of a sub-image. Accepts row (N), column (M) and image parameters which are use in the class call getSubImage(). Class method writelImage() writes the extracted image to a user-named file.

void computeAvgGrayLevel();

Computes the average gray level of an image. User is prompted for the file name of an image to process. ReadImageHeader() function is called and reads header information; a new image is then instantiated and passed to readImage() function which reads the image information. Class call to computeAvgGrayLevel() computes, returns and displays average gray level.

void enlargImage(ImageType&);

Enlarges an image. Receives image object from executeCases() and prompts user for an enlargement factor, calls class method enlargImage(), prompts user for an output file name and calls writelImage() to write new enlarged image to disk.

void shrinkImage(ImageType&);

Shrinks an image. Receives image object from executeCases() and prompts user for a shrink factor, calls class method shrinkImage(), prompts user for an output file name and calls writelImage() to write new reduced image to disk.

void reflectImage(ImageType&);

Reflects an image either vertically or horizontally. Receives image object from executeCases(), calls class method reflectImage(), prompts user for an output file name and calls writeImage() to write new reflected image to disk.

void translateImage(ImageType&);

Translates an image in the - X and + Y directions. Receives image object from executeCases(), calls class method translateImage(), prompts user for an output file name and calls writeImage() to write new reflected image to disk.

void rotateImage(int N, int M, ImageType& image);

Rotates an image between a user-defined -360 and 360 degrees. Receives row (N), column (M) and image object parameters from executeCases() function. User is prompted to enter rotation angle which is converted to radians. Class method rotateImage() processes rotation; function then prompts user for an output file name and calls writeImage() to write new rotated image to disk.

void negateImage(ImageType&);

Computes the negative of an image. Receives image object from executeCases(), calls class method negateImage(), prompts user for an output file name and calls writeImage() to write new negated image to disk.

void useOverloadPlus(ImageType&);

Adds two images. Receives image object from executeCases(), instantiates two image objects to be used in performing addition process; uses class overloaded operators + and =; prompts user for an output file name and calls writeImage() to write new added image to disk.

void useOverloadMinus(ImageType&);

Subtracts two images. Receives image object from executeCases(), instantiates two image objects to be used in performing subtraction process; uses class overloaded operators - and =; prompts user for an output file name and calls writeImage() to write new added image to disk.

B. The Class Functions.**ImageType();**

The default constructor. Sets row, column, gray-level values to zero; sets pixelValue member to NULL

ImageType(int, int, int);

The parameter constructor. Accepts temporary row, column and gray-level parameters then returns current respective values. Sets all pixel values to zero.

ImageType(const ImageType &object);

Implemented and tested but not used.

void getImageInfo(int& rows, int& cols, int& levels);

Returns by reference the number of rows and columns of the image, and the max pixel value.

void getPixelVal(int i, int j, int& val);

Returns the pixel value by reference at (i, j) location. Bounds checking is done in main (but should have been done in this function to provide more robustness to client calls).

void setPixelVal(int i, int j, int val);

Sets the pixel value at location (i, j) to val. Bounds checking is done in main (but should have been done in this function to provide more robustness to client calls).

int computeAvgGrayLevel(const ImageType &object);

Returns the average gray-level of object. For each row then for each column, sums all pixel values in the 2D array then divides by the number of cells (row x columns) to compute an average.

ImageType enlargeImage(int e, ImageType &object);

Enlarges object. Instantiates a temporary Image object to store object parameter by row then by column processing. A second temporary object is instantiated then set equal to object using a row loop from 0 to rows times e and a column loop from 0 to columns times e while the object's rows and columns are set to i/e and j/e, respectively. Returns an image type object.

ImageType shrinkImage(int s, ImageType &object);

Reduces object. Instantiates a temp object and populates with object using two nested for loops then zeroizes the original object. Next, uses row and column nested for loops and increments by s in each loop, cycles through each cell of the temp object and assigns to corresponding object rows and columns divided by s. Then, uses previous loop structure but with column upper bound divided by s, assigns temp cells to corresponding object cells, again with columns divided by s. Lastly, with two nested for loops with upper bounds divided by s but one less than number of rows and columns, respectively, object is assigned to previously instantiated second temp object. Returns second temp object.

void getSubImage(int ULr,int ULc, int LRR,int LRC, ImageType& object)

Returns a sub-image of object. Instantiates a temp image object to which object is assigned then object is zeroized. Two nested for loops, from Upper-Left row (ULr) to Lower-Right row (LRR) and from Upper-Left column (ULc) to Lower-Right column (LRC) then re-populate object with temp object values. Returns object by reference.

void negateImage(ImageType &object)

Negates object. For each row and column of object, object is assigned the absolute value of 255 minus each pixel value of object. Returns object by reference.

void reflectImage(ImageType &object)

Reflects object about the vertical or horizontal axis. Object is assigned to instantiated temp object then zeroized. User is then asked to choose between horizontal and vertical reflection. A switch statement then performs the corresponding choice. Both reflections use two nested for loops – the horizontal's column loop descends from column less one while the vertical's row loop descends from row less one. Reflected image is returned by reference.

ImageType translateImage(int t, ImageType &object)

Shifts an image in the -X and +Y directions by amount t. Assigns object to instantiated temp object. If each row and column index plus the translation amount sum to less than the object's row and column values then the translated values are set in temp and temp is returned by reference.

void rotateImage(int N, int M, float theta, ImageType &object)

Rotates object by a user-specified angle. Object is assigned to instantiated temp object then zeroized. New row and column coordinates are then computed with the following formulas:

$$R = Xc + (i - Xc) * \text{static_cast<int>}(\cos(\theta) + 0.5) - (j - Yc) * \text{static_cast<int>}(\sin(\theta) + 0.5);$$

$$C = Yc + (i - Xc) * \text{static_cast<int>}(\sin(\theta) + 0.5) + (j - Yc) * \text{static_cast<int>}(\cos(\theta) + 0.5);$$

where Xc and Yc are N/2 (row) and M/2 (column), respectively; theta is the angle in radians (computed in Main but should have been done in this function for client robustness), i and j are row and column incrementers, respectively. Casting is performed to convert calculations to an integer while the addition of 0.5 rounds the integers to the nearest integer. Bounds checking is performed to set pixel values to zero for those R and C coordinates that fall outside of object's row and column dimensions. Returns rotated object by reference.

bool inBounds(int R, int C)

Performs row and column bounds checking for rotateImage();.

void operator= (const ImageType &object)

Used when overloaded + and - operators are invoked.

ImageType operator + (ImageType &object)

Used to add two objects.

ImageType operator - (ImageType &object)

Used to subtract one object from another.

~ImageType()

Releases memory used by pixelValue row pointers then pixelValue double pointer upon program termination.

C. Supplied Functions (Supplied by Dr. Bebis and restated here for document completion).

int readImageHeader(char[], int&, int&, int&, bool&);

Reads the header information of an image file. Stores the filename in char[], tests for a valid file. If file is valid, reads the header and test for P5 (PGM) or P6 (PPM) image types. If P5, retrieves row, column and max pixel value information.

int readImage(char[], ImageType&);

Reads in an image from a file. The pixel values are stored in an array and the image width, height, and number of gray-levels are recorded in the appropriate fields. A NOT-PGM exception should be raised if the image file to be read is not in PGM format (image is an object of type ImageType).

writelImage(fileName, image):

Writes an image to a file using a user supplied filename in the appropriate format (image is an object of type ImageType).

IV. Known Issues

When an image is shrunk, any subsequent image processing continues to use the shrunk image. Temporary work around is to exit the program then re-enter. CS302 software engineers are working around the clock to fix the issue.

VI. Lessons Learned

Perhaps the chief lesson learned on this assignment was class robustness. The menu items works primarily on the first image read into the image object. Although this is acceptable, it is restrictive to a client application. More robust coding would allow a client application to read any image for each of the menu items.

getPixelVal() and setPixelVal() functions perform bounds checking in main when it should be performed in the class in order to provide class robustness to a client application.

void rotatelImage() should perform degrees-to-radians conversion rather than the client performing the calculation. Again, this would provide a greater degree of robustness to a client application.

VII. Partner Contributions.

Both partners contributed 50% each to the assignment.