UAH Electrical and Computer Engineering Department
Project01 (3 points) **Version 1.0**

Submit Your Solution to the Canvas Assignment Dropbox by 5:00pm on Friday, February 1, 2019

Important Note

All CPE 212 projects are *automatically graded* in order to provide you timely feedback. So, it is critical that you follow all directions for the preparation and submission of your projects for grading.

Failure to follow the directions may result in zero credit (0 points).

Hint

Match the output of your program to that of the sample solution!!!
Directions for running the sample solution appear on the following pages.
Directions for running the preview script appear on the following pages.

Project01 Goals

A fundamental software engineering skill is the design, implementation, and testing of a software component that must be integrated into a larger software product. In order to do this, the software component must conform to a previously agreed upon interface format. As part of this assignment, you will practice this skill by writing **several functions** that will be integrated into a larger software product provided by the instructor. Along the way you will review basic C++ programming and Linux file management skills required for successful completion of CPE 212.

Project01 Overview

For this project, you will ***complete*** the provided partial C++ program by implementing ***six functions*** that perform simple image processing operations on images stored as 10x10, two dimensional arrays of integers. The image processing operations are:

(1) Get Next Image
(2) Horizontal Flip
(3) Vertical Flip
(4) Transpose
(5) Rotate 90° Clockwise
(6) Rotate 90° Counter Clockwise

You must implement each of these operations as a C++ function. ***Your code for the six functions above must appear in the file named* project01.cpp *in order to compile correctly.*** Remember, spelling and capitalization count in Linux/C++.

The function **main()** has already been implemented for you to provide a common way for everyone to test their code. The function **main()** will scan a sample input file and perform the requested series of image processing operations by invoking your functions as needed.

Prototypes for the six functions are already provided for you in **main.cpp (do not modify main.cpp !!!). All program output is performed by the code in the main.cpp file – do not include any output statements in the file project01.cpp**

The interfaces to each of these functions you must write are described in detail on subsequent pages and in the prototypes listed within main.cpp

Step #1 - Unzipping Project Materials on blackhawk

Use the Chrome/Firefox/Safari browser to access Canvas and download the **Project01_Materials.zip** file into your **Project01** directory. At terminal window prompt, use the unzip utility to uncompress the files. For example, to unzip the files into your current directory:

```
unzip  Project01_Materials.zip
```

Since this project is worth three points, you have been given three input files to test your program.

Step #2 – Create Function Stubs for the Missing Support Functions

Open a Linux terminal window and navigate to the directory containing the **Project01** materials downloaded from Canvas (you should be there already if you just completed Step #1 above).

Use the following command to create the file **project01.cpp** by typing the following at the Linux prompt. Linux is case sensitive so pay attention to upper versus lower case letters.

```
gedit project01.cpp &
```

This command starts the **gedit** text editor running as a background process (**&**). With the editor running in the background, you may use the same terminal window to compile and test your program. Place function stubs for the functions listed above in the file **project01.cpp** and save the file.

In the terminal window, compile your program by typing the word **make** at the prompt and press the Enter key. If you see error messages, use the text editor to make corrections to the function stubs and switch back to the terminal window to type **make** again to recompile your program. Once your program compiles, you may execute it by typing: `./project01  inputfilename`

Now that your program compiles and executes using function stubs, you may implement and test your solution function by function. Start with the functions associated with opening files and loading data.

Running the Sample Solution on blackhawk [shows the desired output for specified input file]

The best description of what your code should do is the Sample Solution for the project.

Run the sample solution by typing the following at **blackhawk** terminal window command prompt where `inputfilename` is the name of one of the provided input files (for example, p01input1.txt).

```
/home/work/cpe212/project01/p01  inputfilename
```

➔ **Your current working directory must contain the input files for this to work.** ➔

Running the Preview Script on blackhawk [analyzes outputs for all provided input files]

Run the preview script by typing the following in a **blackhawk** terminal window command prompt

```
/home/work/cpe212data/project01/preview01.bash
```

This script will run both the **Sample Solution** AND your **project01** executable program on the complete set of input files, and it compares the outputs of the two programs line by line to identify errors in your program's outputs. Make sure that the output of your program exactly matches the output of the Sample Solution.

- ***To use the preview script, your executable must be on blackhawk***
- ***The sample input files must be in your current working directory BEFORE you execute the preview script!!!***

Project01 Hints

Use the function prototypes appearing in **main.cpp** to create a file named **project01.cpp** that contains empty function definitions. Be sure that you can successfully use ***make*** to compile this skeleton solution before adding any additional code.

The **GetNextImage** function is critical – **if this function does not work, then your project will fail every test**. Make sure that **GetNextImage** works before continuing with any other functions.
After **GetNextImage**, implement the simplest operations **FlipHorizontal, FlipVertical,** and **Transpose** ***one at a time***, testing each with the appropriate input file. Rotations may be implemented by combinations of the three simplest operations.

```
struct Record   // Structured data type for storing one line of image data from file
{
    int digit;                          // Digit read from file
    int image[MAXROWS][MAXCOLS];        // Pixel data read from file
};
```

Project 01 Function Specifications

```
void GetNextImage(ifstream& datastream, int counters[], Record& data)
// Reads digit and image data from datastream object storing values in data parameter
// Updates histogram counters to reflect specific digit input from file
// Note: Correct operation of this function is critical!!  If one cannot correctly
//       load images into the array from the file, then one cannot test the
//       image processing operations and your code will fail every test!

void Transpose(int image[MAXROWS][MAXCOLS]);
// Transpose() - must flip the image across the primary diagonal row = column as
// with a matrix transpose
// Parameter image is a two-dimensional array of integers representing the image

void FlipHorizontal(int image[MAXROWS][MAXCOLS]);
// FlipHorizontal() - must flip the image horizontally.  For example,
// column zero exchanged with column N-1, column one exchanged with column N-2, etc.
// Parameter image is a two-dimensional array of integers representing the image

void FlipVertical(int image[MAXROWS][MAXCOLS]);
// FlipVertical() - must flip the image Vertically.  For example,
// row zero exchanged with row N-1, row one exchanged with row N-2, etc.
// Parameter image is a two-dimensional array of integers representing the image

void RotateCW(int image[MAXROWS][MAXCOLS]);
// RotateCW() - must rotate the image 90 degrees clockwise.
// Parameter image is a two-dimensional array of integers representing the image

void RotateCCW(int image[MAXROWS][MAXCOLS]);
// RotateCCW() - must rotate the image 90 degrees counter clockwise.
// Parameter image is a two-dimensional array of integers representing the image
```

Hint: Rotations may be achieved by combining the other operations.

Project01 Sample Input File (p01input1.txt)

 # p01input1.txt -- Test of Transpose Image ← # designates a comment

[smalldata.csv ← Open specified image data file

 # Draw Bars ← b designates drawbar command
b 1

 # Print Original Image ← p designates print image command
p

 # Draw Bars
b 1

 # Test Transpose Image ← t designates transpose command
t

 # Print Image
p

 # Close file
]

One Line from Data File (smalldata.csv) (first integer is digit depicted by pixels – 24x24 array)

7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,84,185,159,151,60,36,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,222,254,254,254,254,241,198,198,198,198,198,198,198,19
8,170,52,0,0,0,0,0,0,0,0,0,0,0,0,67,114,72,114,163,227,254,225,254,254,254,250,229,254,254,140
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,17,66,14,67,67,67,59,21,236,254,106,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,83,253,209,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22,233,255,83,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,129,254,238,44,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,59,249,2
54,62,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,254,187,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,9,205,248,58,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,126,254,182,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,75,251,240,57,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,19,221,254,166,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,203,254,219,35,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,38,254,254,77,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,31,224,254,115,1,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,133,254,254,52,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,61,242,254,254,52,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,254,254,219,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,121,254,207,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0