Name: Tai Bui
CSCI 3731 – Hw05

1. What is the problem with two-dimensional arrays in C/C++?

C/C++ was originally invented to deal with operating systems, thus it does have certain limits in dealing with two dimensional arrays, which often used for storing tables and images purposes. The limits in multi-dimensional array operations arise when users need to pass a multi-dimensional arrays into a function. Users will need to specifically state the size of the multi-dimensional arrays when passing to a function; thus, limit the function's ability to deal with other different size arrays.

2. Describe two ways to work around C/C++'s problems with two dimensional arrays.

The first common way to deal with C/C++'s problems with two dimensional arrays is to use flat array. Users can store data in a one dimensional array with rows laid down end to end. A two-dimensional array with sized width * height, now would be converted into a single array with length width * height. As a result, users now can easily pass an array into a function.

The second most common way to deal with two dimensional arrays in C/C++ is to use "Numerical Recipes trick" which includes the use of a flat array and an array of pointer to the start of each row. This technique allows users to access an element in the array conveniently by simply stating row and column index such as array [row] [column].

3. Is your computer big endian or little endian? Hint: write a small program to find out.

My computer is little endian according to my test.cc. The output displays the int 123456 in the little endian order with 40h, e2h, 1h and 0h.

```
#include <cstdio>

int main(){

    int n = 123456;

    unsigned char* bytes = (unsigned char*)(&n);

    printf("%x %x %x %x\n",(int)bytes[0], (int)bytes[1], (int)bytes[2], (int)bytes[3]);

}
```

```
Tai@TaiBui MINGW64 ~/homework/CSCI3731/hw05
$ g++ -c -Werror test.cc

Tai@TaiBui MINGW64 ~/homework/CSCI3731/hw05
$ g++ -o test test.o

Tai@TaiBui MINGW64 ~/homework/CSCI3731/hw05
$ ./test
40 e2 1 0

Tai@TaiBui MINGW64 ~/homework/CSCI3731/hw05
$
```

Name: Tai Bui
CSCI 3731 – Hw05

4. If you didn’t finish the last programming assignment for reading and writing PPM files with
a flat array, do so now. If I suggested improvements, make them.
Then modify your code to copy the pixels from the flat array into a 2D array of ints. The read
function should return this 2D array instead of the flat array and the write function should
take the 2D array as an argument.
Each pixel has three bytes for the red, green and blue color samples. In the 2D array use one
int for each pixel. Set the first byte of each int to 255 (0xff) and an use the rest for the color samples.
Write a program that reads the image from the last homework and writes a copy of it, like
you did the last time.
If you want to impress me, write another function that modifies the image. For example, flip
or rotate the image, or swap its color channels.
Hints:
_ Use pointer casting to turn an int into an array of four bytes.
_ Use the “Numerical Recipies trick” for storing the two-dimensional array.