Name: Tai Ngoc Bui
CSCI 3731
Hw - 04

1. What is wrong with the following code, and how would you fix it?

```
void swap(int* a, int* b) {
        int* tmp = a;
        a = b;
        b = tmp;
} // end of swap method
```

The function only swaps the address instead of swapping the data values stored in address a and b. The fix would be

```
int tmp = *a;
*a = *b;
*b = tmp;
```

2. What is wrong with the following code, and how would you fix it?

```
char copy[8];
const char* string = "hello";
for(int i=0; string[i] != '\0'; ++i) {
        copy[i] = string[i];
}

printf("%s\n", copy);
```

Problem: The loop will stop when the element is null terminator. Thus, the copy array will not copy the null terminator into its array.

Fix:

```
char copy[8];
const char* string = "hello";
int i = 0;
for(; string[i] != '\0'; ++i) {
        copy[i] = string[i];
}
copy[i] = '\0';
printf("%s\n", copy);
```

3. Suppose you have a function that takes an array as an argument in the usual way, which is declared as:

```
void f (int* array, int length);
```

Suppose you also have an array declared as:

Name: Tai Ngoc Bui
CSCI 3731
Hw - 04

int a[128];
How would you pass a sub-array containing only the third through sixth elements of a to the function f?

Answer: f (a + 2, 4)

4. What is wrong with the following code and how would you fix it?

```
double* allocateArray(int length) {
        double array[length];
        return array;
}
```

Problem: we create an array inside a function and return a pointer to it. When the pointer returns, the array variable is out of scope and the return pointer no longer pointing to the allocated memory.

Fix:

```
double* allocateArray(int length) {
        double* array = new double[length];
        return array;
}
```

5. What is wrong with the following code, and how would you fix it?

```
char string[5];
string[0] = ’h’;
string[1] = ’e’;
string[2] = ’l’;
string[3] = ’l’;
string[4] = ’o’;
printf("%s\n", string);
```

As C strings are just array of chars and the end of a string is marked by an extra char equal to the null character ‘\0’. The char variable string above should have been initialized with length 6 instead of 5, and the last char should be ‘\0’.

```
const char string[6] = {'h','e','l','l','o','\0'};
printf("%s\n", string);
```

6. Write a function that reads PPM format images. The function should return the pixel data in a one dimensional array of unsigned chars, plus the width and height of the image.A PPM image has an ASCII header followed by binary pixel data. The header looks something like this:

```
P6
650 652
255
```

P6 is a string that indicates this is a PPM image. Most file formats start with an identifying string like this. It is called a “magic number”. The next two fields are the width

and height of the image. The last field gives the maximum pixel value. You can expect it will always be 255. At the end of the header is a nn and then the binary pixel data. The image is in color, so there are three bytes (red, green, blue) for every pixel.
Write another function that writes PPM format images, given the information returned from your PPM reader. Write a program that tests your two functions by reading the test.ppm image on BlackBoard and making a copy of it.
Hints:
_ How will you return multiple things from the same function?
_ Think about which function arguments should be const and which should not.
_ You will need to use a combination of formatted and unformatted I/O.
_ Take this one step at a time. First read the header and make sure that code is working right. Then read the pixels.
_ Be careful about that nn.
_ I posted a Java program that reads and displays a PPM image. Note the way you do this in Java may be significantly different from the way you do it in C++. But you can use my Java program to display your PPM image, if your operating system doesn't do it for you. You can also use it to print header and pixel values to compare with your C++ program.