

# CSCI-1200 Computer Science II — Fall 2007

## Homework 1 — Moiré Strings

Before starting this homework, make sure you have read and understood the CSII Academic Integrity Policy.

### Part 1: Short Answer

In your main CSII directory on your laptop, create a subdirectory for homeworks and within that directory create a subdirectory `hw1` for this assignment. Download and open the *plain text* file named `README.txt` and write your solutions for Part 1 in this file.

1. What is the output of the following program?

*Note: While it is clearly possible for you to create a program file containing the code below and then compile and execute the program, you will not have the luxury of doing this on a test. You should practice by studying the code carefully to determine the answers without a computer.*

```
#include <iostream>
using namespace std;

int main() {
    int x=5, y=4;
    float a[3] = { 1.0, 2.0, 3.0 };

    if (x > y) {
        float y = 1.0;
        int a = 6;
        cout << "x = " << x << ", y = " << y << ", a = " << a << endl;
        x = 2;
    }

    cout << "x = " << x << ", y = " << y << ", a[0] = " << a[0] << endl;
    return 0;
}
```

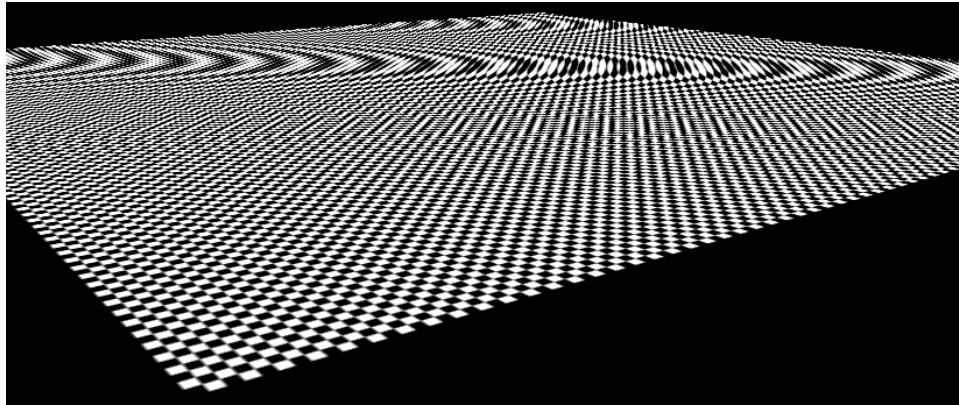
2. For each of the two functions in the following code, give an order notation count of the number of operations it requires in terms of  $n$ . Justify your answer briefly. For extra credit, rewrite the second function to significantly improve its performance, and give the new order notation.

```
// Count the number of entries in an array that have
// both an even index (subscript) and an even value.
int count_evens(int arr[], int n) {
    int count = 0;
    for (unsigned int i=0; i<n; i+=2)
        if (arr[i] % 2 == 0) ++count;
    return count;
}

// Given an array of floats, form an array of sums.
// Entry i is the sum of the values in locations i..n-1.
void subsequence_sum(float arr[], int n, float sums[]) {
    for (int i = 0; i < n; ++i) {
        sums[i] = 0.0;
        for (int j=i; j<n; ++j)
            sums[i] += arr[j];
    }
}
```

## Part 2: Moiré Patterns

A moiré pattern is a visible distortion that can result from a variety of interference conditions. The term comes from the French “moirer” (to water) and is used to describe a rippled, water-like look, which is often a desired artistic effect. The effect can be seen when two geometrically regular patterns (such as two sets of parallel lines or two halftone screens) are superimposed, especially at an acute angle. *Definitions from <http://webster.com> and <http://answers.com>.* Unintentional, distracting moiré patterns can be seen in computer games and other graphical applications when intricate textures (such as a checkerboard below) are displayed in perspective on a computer screen. Various signal processing techniques can be used to reduce the appearance of Moiré patterns.



In this homework you will work with command line arguments, file input and output, and the C++ string class to create *ascii art* with simple moiré patterns! Please read the rest of the assignment before starting to program.

### File I/O

You will read string patterns from a input text file. Each line of the file will have first a string (of 1 or more non-whitespace characters) followed by an integer,  $n$ , which indicates the size of the finished moiré polygon. Here is a sample input file, `in_patterns.txt`:

```
abcde 9
__hi!__ 21
```

### Command Line Arguments

Your program will expect 3 command line arguments. The first is the name of the input file. The second is the name of the output file where the program should output the finished moiré imagery. The third argument will be a string (`square`, `right_triangle`, or `isosceles_triangle`) specifying which type of polygon should be created. Here are examples of valid command lines for your program:

```
moire.exe in_patterns.txt out_square.txt square
moire.exe in_patterns.txt out_right_triangle.txt right_triangle
moire.exe in_patterns.txt out_isosceles_triangle.txt isosceles_triangle
```

You should implement very simple error checking to ensure that 3 arguments are provided and that the input and output file streams are successfully opened. Your program should exit gracefully with a useful error message sent to `std::cerr` if there is a problem with the arguments.

You must follow the specifications for the command line, input file, and output file **exactly** to aid the TAs in grading and ensure you receive full credit. We have provided sample input & output files on the course website, and the validation script on the submission server will also help you check your work.

