CSCI-1200 Data Structures — Fall 2012
Homework 1 — Moiré Patterns

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

Part 1: Short Answer

In your main Data Structures 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 for this homework and write your solutions for Part 1 in this file.

For each sample of code below, give the order notation count of the number of operations it requires in terms of n. Be sure to simplify the your answer. Justify your final answer briefly. For extra credit, rewrite the second function to significantly improve its performance, and give the new order notation.

1.
```
// 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;
}
```

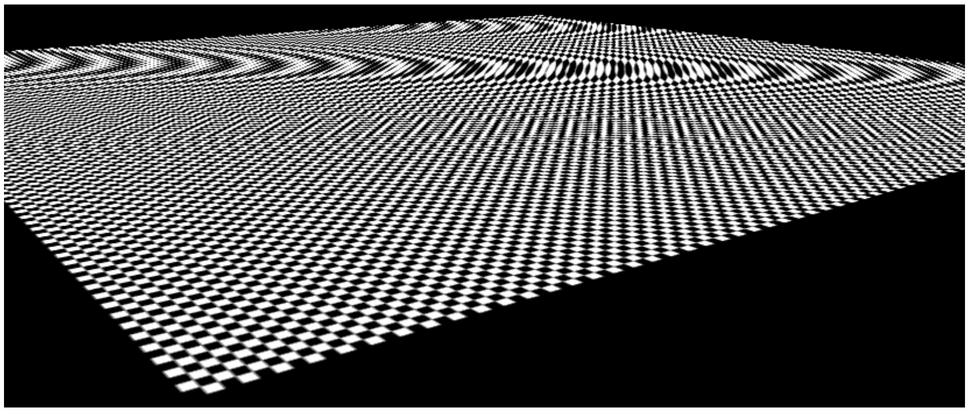
2.
```
// Given an array of floats, form an array of sums.
// Entry i is the sum of the values in locations i..n-1.
void subsequence_sum(float arr[], int n, float sums[]) {
  for (int i = 0; i < n; ++i)  {
    sums[i] = 0.0;
    for (int j=i; j<n; ++j)
      sums[i] += arr[j];
  }
}
```

3.
```
// A somewhat silly example of mutually recursive functions
bool is_even(int n);

bool is_odd(int n) {
  if (n == 0) return false;
  return !is_even(n-1);
}

bool is_even(int n) {
  if (n == 0) return true;
  return !is_odd(n-1);
}
```

4.
```
// A toy counting example
int count = 0;
for (int i = 0; i < sqrt(n); i++) {
  for (int j = 0; j < n; j++) {
    count++;
  }
}
```

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.

Command Line Arguments

Your program will expect 4 command line arguments. The first argument is a 1 or more character string containing the patterns of characters that will be used to fill our ascii art shapes. The second is an integer indicating the desired height of the output shape. The third is a string (`square`, `right_triangle`, or `isosceles_triangle`) specifying which type of polygon should be created. And the fourth is the name of the output file where the program should store the finished moiré image. Here are examples of valid command lines for your program:

```
./moire.exe abcde 9 square abcde_9_square.txt
./moire.exe abcde 9 right_triangle abcde_9_right_triangle.txt
./moire.exe abcde 9 isosceles_triangle abcde_9_isosceles_triangle.txt
./moire.exe ::....... 21 square dots_21_square.txt
./moire.exe '__hi!__' 21 right_triangle hi_21_right_triangle.txt
./moire.exe "|_| " 21 isosceles_triangle blocky_21_isosceles_triangle.txt
```

You should implement simple error checking to ensure that the arguments provided are appropriate and that the output file stream is successfully opened. Your program should exit gracefully with a useful error message sent to `std::cerr` if there is a problem with the arguments. Note in the last 2 examples that when the pattern includes special characters (e.g., `!`) or whitespace it may be necessary to put the pattern string in single or double quotes or use backslash (\) to ensure the data is successfully passed into your program. Your program doesn't need to do any extra processing for these examples to work.

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

Pretty Pictures!

The output from your program will be stunning *ascii art*. The output shape will consist of a border outline of asterisks with an overall height of each polygon *including the border* is equal to the 2nd argument on the command line.

The interior of each polygon is filled with the string pattern from left to right. When you run out of characters in the string pattern, start again at the beginning of the string pattern, leaving no white space. When you reach the right edge of the polygon remember where you left off so that you can start with that character on the next line.

Here is the output (6 different files) that should be produced by your program for the sample command line arguments above:

`abcde_9_square.txt`

```
*********
*abcdeab*
*cdeabcd*
*eabcdea*
*bcdeabc*
*deabcde*
*abcdeab*
*cdeabcd*
*********
```

`abcde_9_right_triangle.txt`

```
*
**
*a*
*bc*
*dea*
*bcde*
*abcde*
*abcdea*
*********
```

`abcde_9_isosceles_triangle.txt`

```
        *
       *a*
      *bcd*
     *eabcd*
    *eabcdea*
   *bcdeabcde*
  *abcdeabcdea*
 *bcdeabcdeabcd*
*****************
```

`dots_21_square.txt`

```
*********************
*::.......::.......:*
*:.......::.......::*
*.......::.......::.*
*......::.......::..*
*.....::.......::...*
*....::.......::....*
*...::.......::.....*
*..::.......::......*
*.::.......::.......*
*::.......::.......:*
*:.......::.......::*
*.......::.......::.*
*......::.......::..*
*.....::.......::...*
*....::.......::....*
*...::.......::.....*
*..::.......::......*
*.::.......::.......*
*::.......::.......:*
*********************
```

`hi_21_right_triangle.txt`

```
*
**
*_*
*_h*
*i!_*
*___h*
*i!___*
*_hi!__*
*__hi!__*
*__hi!___*
*_hi!____h*
*i!____hi!_*
*___hi!____h*
*i!____hi!___*
*_hi!____hi!__*
*__hi!____hi!__*
*__hi!____hi!___*
*_hi!____hi!____h*
*i!____hi!____hi!_*
*___hi!____hi!____h*
*********************
```

`blocky_21_isosceles_triangle.txt`

```
                    *
                   *|*
                  *_| *
                 *|_| |*
                *_| |_| *
               *|_| |_| |*
              *_| |_| |_| *
             *|_| |_| |_| |*
            *_| |_| |_| |_| *
           *|_| |_| |_| |_| |*
          *_| |_| |_| |_| |_| *
         *|_| |_| |_| |_| |_| |*
        *_| |_| |_| |_| |_| |_| *
       *|_| |_| |_| |_| |_| |_| |*
      *_| |_| |_| |_| |_| |_| |_| *
     *|_| |_| |_| |_| |_| |_| |_| |*
    *_| |_| |_| |_| |_| |_| |_| |_| *
   *|_| |_| |_| |_| |_| |_| |_| |_| |*
  *_| |_| |_| |_| |_| |_| |_| |_| |_| *
 *|_| |_| |_| |_| |_| |_| |_| |_| |_| |*
*****************************************
```

These output test files are available on the course webpage.

Even More Fun!

For extra credit, try other shapes. Write about your extensions in your *plain text* `README.txt` file. Paste sample command lines and output for your extra credit into your README.txt file.

Submission Details

Do all of your work in a folder named `hw1` inside of your Data Structures homeworks directory. Use good coding style when you design and implement your program. Organize your program into functions: don't put *all* the code in `main`! Use good variable and function names. Be sure to make up new test cases and don't forget to comment your code! Use the provided template `README.txt` file for notes you want the grader to read. **You must do this assignment on your own, as described in the "Academic Integrity for Homework" handout. If you did discuss the problem or error messages, etc. with anyone, please list their names in your `README.txt` file.** Prepare and submit your assignment as instructed on the course webpage. Please ask a TA if you need help preparing your assignment for submission or if you have difficulty writing portable code.