1 Introduction

This lab is about `while` loops. There's nothing to turn in.

2 Puzzles

Instructions: For this section, work together at a whiteboard. Without using your computer, try to figure out exactly what each of the following code snippets will print. Once you've finished solving every problem, then write the code for each one and check yourself. Figure out what happened for each of your mistakes.

1.
   ```
   int n = 0;
   while (n < 10) {
       cout << 2 * n << endl;
       n = n + 1;
   }
   ```

2.
   ```
   int n = 7;
   while (n >= -1) {
       cout << n / 4 <, endl;
       n = n - 1;
   }
   ```

3.
   ```
   int n = 0;
   int m = 10;
   while (n < m) {
       cout << n << endl;
       n = n + 1;
       m = m - 1;
   }
   ```

4.
   ```
   int i = 25;
   while (i < 50) {
       i = i * 2
   }
   if (i > 50) {
       i = i - 4;
   }
   cout << i << endl;
   ```

5.
   ```
   int i = 1;
   while (i < 50) {
       if (i % 5 == 0) {
           cout << i << endl;
       }
   }
   ```

6.
   ```
   int i = 1;
   while (i < 100) {
       cout << i << endl;
       i = i * 2;
   }
   ```

7.
   ```
   int n = 0;
   while (n < 20) {
       cout << n << endl;
       if (n % 2 == 0) {
           n = n + 3;
       } else {
           n = n + 2;
       }
   }
   ```

3 Coding Practice

You may try these problems in any order after you have finished the puzzles. The first two are easier, but the **third question is particularly important** for the next homework assignment.

1. Write a program that asks the user for a positive integer `n`. Compute and print the sum $1 + 2 + 3 + \ldots (n - 1) + n$.
2. Write a program that asks the user for a positive integer `n`. Compute and print the value of *n factorial*: $1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n - 1) \cdot n$.
3. Use a `while` loop to check user input. More specifically, write a program that asks the user for an integer between 5 and 12. If they give an int outside of that range, display an error message and ask them to try again. Continue displaying an error and retrying until the user's input is valid. This is a programming pattern that shows up a lot.

4 Just for Fun

What does this compute?

```
int sign = 1;
double output = 0;
int n = 0;
while (n < 1000) {
    output = output + sign / (2.0 * n + 1.0);
    sign = sign * -1;
    n = n + 1;
}
cout << "The special number is: " << 4 * output << endl;
```

Isn't that cool? How do you make it more accurate? You can use

```
cout.precision(17);
```

if you want to see as many digits as possible.