Name: _____

Today's Date: _____

Today's Goals

- Explain the benefits and limitations of empirical testing
- Translate (nested) for loops to summations
- Find closed forms for summations using common patterns

Today's Question(s)

What are some ways a program can be "good"?

We'd like to strike a balance between printing a lot of handouts (using lots of paper!) and giving you a structured starting point for your notes. Please let us know if you'd like us to continue printing a full packet for you, or if you'd rather get a smaller packet with just the worksheets for the day.

[ ] Please print a full packet for me

[ ] Please only print worksheets and class activities for me.

Lingering Questions

Counting `for` loops

- Why?

```
int main() {
    int total = 0;
    for (int i=1; i < 5; ++i) {
        total += 1;
    }

    cout << total << endl;
}
```

Comparing Algorithms

Interpreting Empirical Data

We can measure...

Empirical Data + What?

Guidelines

These "rules" work most of the time:

Closed Forms for Common Summations

$$\sum_{i=0}^{n-1} 1 = \sum_{i=1}^{n} 1 = n$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Closed Forms for Common Summations

$$\sum_{i=1}^{log_m n} m^i = \frac{m}{m-1}(n-1)$$

For example,

$$\sum_{i=1}^{log_2 n} 2^i = 2n - 2$$

Closed Forms for Common Summations

$$\sum_{i=1}^{n} \frac{1}{i} = H(n) \approx \ln n$$

Exercise 1

```
int main() {

    int data[N];

    for (int i=1; i <= N; ++i) {
        data[i] += 1;
    }

    return 0;
}
```

Metric:

Summation:

Closed Form:

Exercise 2

```
int main() {
    int data[N];

    for (int i=1; i <= N; ++i) {
        for (int j=1; j <= i; ++j) {
            data[i] += j;
        }
    }

    return 0;
}
```

Metric:

Summation:

Closed Form:

Exercise 3

```
int main() {
    int data[N];

    for (int i=1; i <= N; ++i) {
        for (int j=1; j < i; j += 2) {
            data[i] += j;
        }
    }

    cout << total << endl;

    return 0;
}
```

Metric:

Summation:

Closed Form:

Exercise 4

```
int main() {
    int data[N];

    for (int i=1; i < N; i *= 2) {
        for (int j=1; j < i; j += 2) {
            data[i] += j;
        }
    }

    cout << total << endl;

    return 0;
}
```

Metric:

Summation:

Closed Form: