Name: _____

Section: _____

Today's Goals:

- Identify when a *reference* should be used in C++ code
- Identify when *const* should be used in C++ Code
- Model `const` and references in the CS70 memory model.

Today's Question(s)

How do we know that the *compiler* must be responsible for identifying any errors that have to do with the *type* of variables?

What data are you visualizing for homework 2?

Lingering Questions

What is still unclear after today's class?

Redundant?

Instead of creating a new variable to hold our result and then return it, why not modify the variable we get directly?

Exercises

References

A **reference** is ...

Example: `float& x = y;` is read...

Using references

- Key point: When you do something to a reference, you're acting on...
- When an existing reference is used, we read it...

const

What if we have a value that should *not* change?

- Why might we do that?
- How do we say that in C++?
- How do we model that?

const + References?

What happens if we combine `const` and references?

Exercise 1

```
void triple(int multiplier)
{
   int product = 3 * multiplier;
}

int main()
{
    int myInt = 14;
    triple(myConstant);

    cout << myInt << endl;

    return 0;
}
```

Exercise 2

```
void cube(int base)
{
    int outcome = base * base;
    outcome = outcome * base;
}

int main()
{
    int myInt = 3;
    cube(myConstant);

    cout << myInt << endl;

    return 0;
}
```