CSI 201
Functions - Using vectors

1. Functions can be used with vectors.

2. Passing a vector is usually done in a special way. We're going to just start by saying "that's the way it is done" without much explanation. Next class we'll clarify why it happens that way.

3. Let's begin with some sample code:

```
double average(const vector<double> &grades) {
   double sum = 0;
   for(unsigned i = 0; i < grades.size(); ++i) {
     sum = sum + grades.at(i);
   }
   return sum / grades.size();
}
```

4. For the most part, passing a vector to a function is the same as usual. However, it is common to use these extra pieces in the function definition. You should notice that the `const` and the `&` are atypical. Let's just imagine that we always put them there, for now. The const goes before the word vector and the & goes after the > but before the variable name.

5. What does the function call look like? Here's a sample main:

```
int main() {
   vector<double> grades = {100, 70, 80}; //initialize a vector
   cout << average(grades) << endl; //should output 83.333333
}
```

6. Now let's try some more! Write a function to output every element of a vector!

7. Write a function that takes in a vector and computes the sum of all the elements in the vector.

8. Write a function that takes in a vector and finds the maximum in the vector.

9. Write a function that takes in a vector and finds the minimum in the vector.

10. Write a function that finds the average value of all the elements but with a special rule. Any value that is below 50 is treated as if it is 50 instead. So the average of 0, 0, 0 is just 50. The average of 0, 60 is 55.

11. Write a function that determines if the elements are strictly increasing. For example: [50, 60, 70] is always increasing. However, [60, 70, 60] increases at first and then does not. The function might have the prototype:

```
bool is_increasing(const vector<double> &grades);
```

12. Write a function that computes the average of a vector of doubles. However, if the values are increasing (feel free to use `is_increasing` from above) then simply return the value of the last element.