CSI 201
Sums, max and min over whole vectors!

1. Quickly want to reiterate three things about vectors!

   (a) We can use assignment (=) to make a copy of one vector to another. They'll become the same size and have the same elements after. Imagine `a=b;`. If a and b are integers, then a becomes a copy of b. The same is true if they're both vectors.

   (b) We can change the size of a vector using `.resize(int)`. We can change them to have 0 size or any size we like. This is often done to drop the last element of a vector or to make a vector a certain size before we fill it with data. If s is a vector, I can write: `s.resize(12)` to change the number of elements in s to 12, regardless of whether it started with 27 elements, or 3 elements or 0. I can also, change the size to 0, to "remove" all elements from the list.

   (c) We can even create vectors with a given size when we make them. We simply add `(int)` at the end. For example, to make a list of strings with 100 elements we can write: `vector<string> my_names(100);`

2. Let's start with sample code from last time:

```
#include <iostream>
#include <cstdlib>
#include <vector>
using namespace std;
int main() {
  int user_seed;
  cin >> user_seed;
  srand(user_seed);
  vector<int> myrands;
  for(int i = 0; i < 20; ++i) {
     myrands.push_back( rand() % 2 );
  }
  //new code snippet from below will go here
}
```

3. At this point, it might be nice to see, "how many 1s are in this list?" Let's do an exercise. I'll say 20 numbers, (0s and 1s) out loud. When the list is done, tell me what the total number of 1s might be. Then, write down a description of how we figured it out on paper.

4. Would this process change, if we were asked to count the number of 0s?

5. Okay, so let's write some code that actually puts this strategy into practice. Remember, we don't do this all at once. We do this gradually as we inspect each element of the vector.

```
int sum = 0; //this 0 is important this time!
for(unsigned i = 0; i < myrands.size(); ++i) {
  //if we count 0s, I might need to be slightly
  //more clever here or (at worst) use an if
  sum = sum + myrands.at(i);
}
//when we get here, we know the # of 1s in the list
```

6. With this code, we now have the # of 1s in the list. So, we can talk about the percentage of 1s compared to the percentage of 0s or, well, anything!

```
//we could try this, but it gives integer arithmetic
int pct_of_1s = sum / myrands.size();
```

7. What does the above code most likely output?

8. Let's try again:

```
//option 1 is to implictly cast as a double first.
double pct_of_1s = sum;
pct_of_1s = pct_of_1s / myrands.size();

//option 2 is to explicitly cast to a double
double pct_of_1s = (double)sum / myrands.size();
```

9. Often, we will use option 1, because we believe that if we think of types as only appearing in variable declarations, it might make code less confusing overall.

10. Okay, let's try to find the sum of an integer array of grades. So let's say we have a list of grades `vector<int> grades;`, that was filled in by the user. We don't know how many we have, but we want to figure out what the average grade might be. Write some code to compute the sum of all the grades first. Then, compute the average.

11. Can we find the maximum value? Start by assuming the first value is the maximum. Then walk through the list to see if any values are bigger and replace the maximum! Try it!

12. We can do the same to find the minimum!

13. We can write a standard deviation once we have the average. If x_a is the average, x_i is the ith element of the list and N is the number of elements, the standard deviation is: $\sigma = \sqrt{\sum_{i=1}^{N} \frac{(x_i - x_a)^2}{N-1}}$ The $\sum$ means the sum of all those things.

14. How might we find the median? This turns out to be significantly harder. It requires us to do a lot more work! What kind of algorithms might we use? This, surprisingly, might be something that requires more tools than we are prepared to use at the moment.