Filling & Shuffling

You can automatically fill a `vector` with any value you like when you create it by using one of the constructors. To fill a `vector` with a sequence of **different values** when that sequence is dependent on the loop counter, use a counter-controlled loop like this:

```
const int kNumbers = 50;
vector<int> lotter(kNumbers);     // sized vector
for (int i = 0; i < kNumbers; ++i)
{
    v.at(i) = (i + 1);
}
```

Once you have the `vector` filled, its time to randomly rearrange the elements, a process called **shuffling**. The best algorithm, known as the **Fisher-Yates** or **Knuth** shuffle, works like this:

1. Take the **last** ball in the `vector` (or card in the deck), and exchange it with any other ball. After this exchange, this ball will never be swapped again. It will also be guaranteed **not** to be itself.
2. Then, take the next-to-last ball, and exchange it with any of the remaining balls.

Continue on until the first ball has been swapped. Here's the shuffle algorithm in code:

```
for (size_t i = lottery.size(); i > 0; --i)
{
    size_t j = rand() % i;

    int temp = lottery.at(j);
    lottery.at(j) = lottery.at(i);
    lottery.at(i) = temp;
}
```