vector Initialization

A newly-created `vector` is **empty**; it contains no elements. To create a **sized** `vector`, specify the initial size (in **parentheses**) when the `vector` is created. For example, to create a `vector` that initially holds fifteen elements, write:

```
vector<int> iVec(15);
```

This specifies the initial size; you **may** add additional elements later. All elements are **default-initialized**. For primitive types, such as `int`, that means they are set to `0`. For class types, such as `string`, each element is constructed by implicitly calling its **default constructor**.

You may want to initialize all elements with **a value other than zero**. Suppose, for example, you want five copies of the `string` `"(none)"` or twenty copies of the `int` value `137`. To do this, **specify both** the number of elements, and default value for each element elements like this:

```
vector<int> v137(10, 137);
vector<string> vNone(5, "(none)");
```

This syntax is **only** legal when initially **creating** a `vector`. You **must use parentheses**; if you use braces, something else will happen.

List and Copy Initialization

In C++11 (or later), you can specify a **initial list** of values. Write the values, separated by commas, and**surrounded by braces**. **This doesn't work in C++ 98**.

```
vector<string> months{"Jan", "Feb", "Mar"};
```

Lastly, you can initialize one `vector`, from an existing `vector`. The new `vector` is a **completely independent copy** of the original, not an alias, as in Java. Here's how:

```
vector<int> v1{...};
. . .
vector<int> v2(v1);
```