CSCI 2100

Vectors
Announcements
- Test on Tuesday
- Review Monday
- HW due today
- Lab due today
- Next HW: cover Vectors, posted mid-next week
Last time: Queues

Simple data structure
limited access, fast
O(1) time for everything
Today: Vectors

Similar to lists in Python
(we saw them in lab this week)

Our implementation:
-array based

Main functions - see STL, because there are a lot of them!

```cpp
private:
  unsigned int size;
  int capacity;
  Object* data;

  data = new Object[capacity];
  data[i]
```
To think through:

Vectors char>

myvec. insert (2, 'c');

How to insert if we don't want to lose data?

myvec: size = 5

capacity = 8

data: [x, y, z, a, b]

Goal: [x, y, c, z, a, b]

for (int i = size; i > index; i--)

data[i] = data[i-1];

data[index] = element;

(throw error if index >= size)
Similarly, erase:

my vec. erase(3);

Underneath:

```
data: [5 12 11 * 6 3 1 1 1 1]
```

Size = 6
Capacity = 11

```cpp
out << myvec[3]
```

Throw error if i >= size
for (int i = index; i < size - 1; i++)
```
data[i] = data[i + 1];
size--;
```
Another issue: what if we exceed the capacity?

Suppose in `insert`,

```c
// create a bigger array
object * old = data;
object * capacity *= 2;
data = new object[capacity];
// copy data over
for (int i = 0; i < size; i++)
data[i] = old[i];
// delete old array
delete[] old;
// continue w/insert
```
Finally, don't forget housekeeping!

- already saw destructor
- copy cons & operator =
  need to allocate a new array & copy data