CS 162
Intro to CS II

Vectors and
Standard Template Library
Odds and Ends

• Assignment #4 posted
• No office hours Friday
• No class next Friday (go to the Expo!!)

extra credit
More Vectors...

- What is the size vs. capacity?
- What is resize() vs. reserve()?
- How does push_back() work?
- What if you needed to make a resize() and reserve()?
- How would this change push_back()?
```cpp
#include "./vector.hpp"
#include <vector>
#include <iostream>

// We do not want to include either stmt. We wouldn't
// be able to compare our vector template to the Standard
// using namespace std;
// using std::vector;
using std::cout;
using std::endl;

int main (){
    vector<int> v;  // Our vector class
    std::vector<int> stdv; // Standard vector

    stdv.push_back(23);
    cout << "cap " << stdv.capacity() << endl;
    cout << "size " << stdv.size() << endl;
    cout << "ele 0 " << stdv.at(0) << endl;
    stdv.resize(10);
    // stdv.reserve(10);
    cout << "cap " << stdv.capacity() << endl;
    cout << "size " << stdv.size() << endl;
    stdv.push_back(100);
    cout << "cap " << stdv.capacity() << endl;
    cout << "size " << stdv.size() << endl;
    cout << "ele 1 " << stdv.at(1) << endl;
}
```
Demo...

```c++
1 #include <stdlib.h>
2 #include <cassert>
3 template <class T>
4 class vector {
5   private:
6     T *v;
7     int s;
8     int cap;
9   public:
10     vector()
11         s=0;
12         cap=0;
13         v=NULL;
14     }
15     ~vector(){
16         delete [] v;
17     }
18     int size() {
19         return s;
20     }
21     int capacity() {
22         return cap;
23     }
```
```cpp
void resize(int new_size) {
    T *temp;
    temp = new T[new_size];
    for(int i=0; i<s; i++)
        temp[i]=v[i];
    delete [] v;
    v=temp;
    cap=new_size;
}

void push_back(T ele) {
    if(cap==s) {
        T *temp;
        s++; cap++;
        //standard says we only need to increase cap by one!
        temp = new T[cap];
        for(int i=0; i<s-1; i++)
            temp[i]=v[i];
        delete [] v;
        v=temp;
        v[s-1]=ele;
    }
    else if(s < cap) {
        v[s++]=ele;
    }
```
Demo...

```cpp
#include "./vector.hpp"
#include <vector>
#include <iostream>

// We do not want to include either stmt. We wouldn't
// be able to compare our vector template to the Standard
// using namespace std;
// using std::vector;
using std::cout;
using std::endl;

int main (){
  vector<int> v;   // Our vector class
  std::vector<int> stdv;  // Standard vector
  // Compare operation of our vector to std
  v.push_back(23);
  cout << "cap " << v.capacity() << endl;
  cout << "size " << v.size() << endl;
  // cout << "ele 0 " << v.at(0) << endl;
  stdv.push_back(100);
  cout << "cap " << stdv.capacity() << endl;
  cout << "size " << stdv.size() << endl;
  // cout << "ele 1 " << v.at(1) << endl;
```