CS 161
Intro to CS I

Strings, cin, and errors
Finish C++ String Demo

• What does cin do when reading...
  – Int/Floats
  – Strings
• What does getline do?
```cpp
#include <iostream>
using namespace std;

int main() {
    int x;
    float f;

    // Not a good way to handle errors because 4.6, 4t, etc. will work
    do {
        if (cin.fail()) {
            cin.clear(); // reset failbit
            cin.ignore(256, '\n'); // ignore 256 chars or until newline
        }
        cout << "enter int: ";
        // cin ignores leading whitespace and reads until char not of
        // the type specified or whitespace (newline, space, tab, etc.)
        cin >> x;
        cout << "value of x: " << x << endl;
    } while (cin.fail()); // fail bit set if it doesn't get type expected

    cout << "enter a float: ";
    cin >> f;
    cout << f << endl;

    return 0;
}
```
In-Class Exercise #5

• How would you determine if user entered a good positive int without using cin.clear() and cin.ignore()?
```cpp
#include <iostream>
#include <string>  // c++ strings
#include <cstdlib>  // atoi()  

using namespace std;

int main() {
    int x;
    float f;
    string s;  // create a string object
    bool bad;  // create a flag to indicate bad or good data

    // cin and getline difference: cin stops reading at any whitespace
    // and stays at the whitespace where it left off reading, getline
    // reads until a newline is encountered and moves past the newline
    cout << "enter a string: ";
    // cin >> s;  // read a string of chars from the user until whitespace
    getline(cin, s);  // read chars from the user until a newline
    cout << "length: " << s.length() << endl;
    cout << "First char: " << s.at(0) << endl;  // 0 is 1st location
    cout << "The whole string: " << s << endl;

    do {
        // Insert
        // Insert
        // Insert
        // Insert
        // Insert
    }
    while (false);  // insertion point here

    return 0;
}
```
do {
    bad=false;  //assume the user is not going to supply bad data
    cout << "enter int: ";
    cin >> s;  //read data as a string, so it will never fail.
    //check that all the characters are 0-9 for a positive int
    for(int i=0; i<s.length(); i++)
        if(!(s.at(i)=='0' && s.at(i)!='9')) {
            bad=true;  //if a char in the string is not 0-9, not positive int
            break;  //break out of the for loop when seeing bad data
        }
} while(bad);  //while the user entered bad data re-prompt them

//after we know input is good, then we can change it to integer, but
//atoi takes a c-style string as input, not a c++ style string
x=atoi(s.c_str());
cout << x << endl;

cout << "enter a float: ";
cin >> f;
cout << f << endl;
return 0;
More about **break, exit, and return**

- **break** – used with switch and loops, breaking out of the closest associated case or loop (for, while, or do while). **This statement can only occur in a loop or case**, otherwise the compiler yells!

- **return** – leave the current function, which exits the program when in the main() function. You can put this **anywhere inside any function**, otherwise the compiler yells!

- **exit()** – exit the entire program, no matter where this is encountered. You can put this **anywhere inside any function**, as long as you include `<cstdlib>`, otherwise the compiler yells!