stringstream

#include <sstream>

The stringstream class is extremely useful in parsing contest input. It associates a string object with a stream allowing you to read from the string as if it were a stream (like cin).

The basic methods for stringstream are:

- `clear ()` -- clears the error bits in the stringstream object so that you can use it again. You need to call this each time you start parsing a new string with the stringstream object. If you don't, the object could still be in a bad state and you may not be able to read anything from the stringstream (even if you change its associated string).
- `str ()` -- returns the string associated with the stringstream object.
- `str (s)` -- associates the string s to the stringstream object.
- `operator<<` -- add a string to the stringstream object.
- `operator>>` -- read something from the stringstream object.

As an example, suppose you were told that each test case will be a single line of input with a series of numbers on it. You can't use a bunch of `cin`'s to read the numbers because you don't know how many of them there are. You have to read off the line using `getline` and then parse the input line. One way would be to scan through the input line looking for spaces, call the `substr` method of the string class to get the substring for each number, convert it to a C string (the member method `c_str()`) and then call atoi on that. A far easier way is to use stringstream like this:

```
#include <iostream>
#include <sstream>
#include <string>

using namespace std;

int
main (void)
```

```
{
        stringstream ss;
        int foo;
        string inp;

        while (getline (cin, inp)) {
                ss.clear ();
                ss.str ("");
                ss << inp;

                while (ss >> foo) {
                        cout << foo << endl;
                }
        }

        return 0;
}
```

Here's a more difficult example. Suppose your input is a series of pairs of the form *(int,str)*. Pairs are separated in the input by spaces. If there are no embedded spaces in a given pair, you can read and parse them by:

```
#include <iostream>
#include <sstream>
#include <string>

using namespace std;

int
main (void)
{
        stringstream ss;
        string inp, w;
        int num;
        char ch;

        while (cin >> inp) {
                inp.resize (inp.size () - 1);
                ss.clear ();
                ss.str ("");
                ss << inp;
                ss >> ch >> num >> ch >> w;
                cout << num << " <--> " << w << endl;
        }
```

```
        return 0;
}
```

Note in the above examples that `clear()` and `str ("")` were always called before using the stringstream object. These two method calls are needed to reset the internal state of the stringstream object.

`clear()` is needed to reset the internal error flags. If this is not called, the stringstream object may think it's in an error state (such as EOF), even if the underlying string has been changed.

`str ("")` is needed to clear out the old string. By calling the stream insertion operator, you are *adding* to the internal string kept by the stringstream object. This feature is useful if you are building up a string from variables and other strings, but can cause your programs to reparse old data.