Big numbers

Bruce Merry

Big numbers in C++

- No equivalent to BigInteger from Java
- Do it manually by storing the digits
- Go back to primary school drills
- Operator overloading makes life pleasant
  - a + b instead of a.add(b)

Implementation tips

- Store the digits in reverse order
  - d[0] is always units, d[1] is always tens etc.
- Keep track of how many digits there are
  - Remember to prune leading zeros
- Helps to have get and set methods to return a digit and handle past-the-end
- Can inherit from vector<int>
- Sign bit only if needed

Example: get and set

```
class big : public vector<int> {
  int get(int d) {
    if (d >= size()) return 0;
    return (*this)[d];
  }
  void set(int d, int v) {
    if (d >= size()) resize(d+1);
    (*this)[d] = v;
  }
};
```

Example: add

```
...
  big operator +(const big &b) const {
    int c = 0;
    big ans;
    for (int i = 0; c || i < size()
         || i < b.size(); i++) {
      c += get(i) + b.get(i);
      ans.set(i, c % 10);
      c /= 10;
    }
  }
```

Example: <

```
...
  bool operator <(const big &b) const {
    // Make sure no leading 0's!
    if (size() != b.size())
      return size() < b.size();
    return lexicographical_compare(
        rbegin(), rend(),
        b.rbegin(), b.rend());
  }
```

Optimisation

- Work base 10^k for $k > 1$
- Work base 2^k and convert for I/O
  - Usually not worth the effort
  - Sometimes no I/O is required
- Be careful not to overflow