Intersection, Union, Multisets

- Finding the intersection of two sets requires examining all the elements of one set and determining if these are in the other
  - How can we “examine” all the elements of a MultiSet?
    - What mechanism exists for accessing individual elements?

- Finding the union of two sets requires examining all the elements of both sets
  - Why iterate over one set for intersection, but two for union?

- Iteration is done in MultiSet, each element processed in an MSApplicant object
  - Code for processing separate from iteration

Anatomy of Intersection

- What we want to write:

  MultiSet a,b,c;
  // fill a, b with values
  c = intersection(a,b);

- This has problems for two reasons
  - We cannot really assign one MultiSet to another
  - The problem has been pushed down a level into the function intersection, we still have to write that

  First: why doesn’t copy work (see copyprob.cpp)
  - Assignment of objects in C++: copy private state
  - What if private state is a pointer?

Shallow copy

- MultiSet object state: two pointers
  - These point to a linked list
  - Assigning a multiset copies pointers

- Given the picture, what happens:
  - a.clear();
  - b.insert("Fe");

- Making a deep copy:
  - Overload assignment operator =
  - Copy the linked list nodes, not just pointers

Anatomy of Intersection (continued)

- Idea for creating intersection of MultiSets a and b

  foreach value in a
    if (b.count(value) != 0) add to intersection
  - Is it possible to add the same value more than once?
  - What should intersection set be initially?

  Here’s the real code for this:
  MultiSet a,b,c;
  // fill a, b with values
  MSIntersect inter(c);
  inter.doIntersect(a,b);

  How does this fill c with the result? MultiSet c must be accessible in the MSIntersection object
MSIntersect, the interface

class MSIntersect : public MSApplicant
{
public:
    MSIntersect(MultiSet& ms);
    void doIntersect(const MultiSet& lhs, const MultiSet& rhs);
    virtual void apply(const string& word, int count);
private:
    MultiSet& mySet;
    const MultiSet * myTempSet;
};

mySet is bound at construction time, not a copy, cannot change
myTempSet is specified when doIntersect is called, can change

MSIntersect, the implementation

MSIntersect::MSIntersect(MultiSet& ms)
    : mySet(ms), myTempSet(0)
{}

void MSIntersect::doIntersect(const MultiSet& lhs, const MultiSet& rhs)
// post: mySet is intersection of lhs and rhs
{
    myTempSet = &lhs;
    rhs.apply(*this); // what is this? What is this?
}

void MSIntersect::apply(const string& word, int count)
// post: mySet is intersection of myTempSet and applied set
{
    if (myTempSet->count(word) != 0) // what is myTempSet?
    { mySet.insert(word); // what is mySet?}
}

How does Union work?

- Idea for creating union of MultiSets a and b
  
  foreach value in a
  add value to union
  
  foreach value in b
  add value to union

  Is it possible to add the same value more than once? Why?

  void MSDoodle::apply(const string& word, int count)
  {
      if (mySet.count(word) == 0) mySet.insert(word);
  }

Understanding intersection and union

- Here's a scenario for two MultiSet objects

  MultiSet a, b;
  a.add("apple");
  a.add("cherry");
  b.add("apple");
  b.add("cherry");
  MSIntersect inter(b);
  b.doIntersect(a, a);

  What is stored in MultiSet b? Why?