```
#include <string>
using namespace std;

#include "tvector.h"
#include "tset.h"

class MemberCheck
{
   public:

    tvector<string> set2vector(const tset<string>& s)
    // post: return vector with same elements as s
    {

        Iterator<string> * it = s.makeIterator();
        tvector<string> result;
        for(it->Init(); it->HasMore(); it->Next()){
            result.push_back(it->Current());
        }
        return result;
    }

    tset<string> vector2set(const tvector<string>& v)
    {
    // post: return set with same elements as v
        tset<string> set;
        for(int k=0; k < v.size(); k++){
            set.insert(v[k]);
        }
        return set;
    }

    tset<string> intersection2(const tset<string>& s1,
                               const tset<string>& s2)
    // post: return set with elements in intersection of s1 and s2
    {
        tset<string> result;
        Iterator<string> * it1 = s1.makeIterator();
        Iterator<string> * it2 = s2.makeIterator();
        it1->Init();
        it2->Init();
        while (it1->HasMore() && it2->HasMore()){
            if (it1->Current() < it2->Current()){
                it1->Next();
            }
            else if (it1->Current() > it2->Current()){
                it2->Next();
            }
            else {
                result.insert(it1->Current());
                it1->Next();
                it2->Next();
            }
        }
        return result;
    }

    tset<string> intersection(const tset<string>& s1,
                              const tset<string>& s2)
    // post: return set with elements in intersection of s1 and s2
    {
        tset<string> result;
        Iterator<string> * it = s1.makeIterator();
        for(it->Init(); it->HasMore(); it->Next()){
            if (s2.contains(it->Current())){
                result.insert(it->Current());
            }
        }
        return result;
```



```
    }

    tvector<string> whosDishonest(const tvector<string>& club1,
                                  const tvector<string>& club2,
                                  const tvector<string>& club3)
    {

        tset<string> s1 = vector2set(club1);
        tset<string> s2 = vector2set(club2);
        tset<string> s3 = vector2set(club3);

        tset<string> result;
        result.insert(intersection2(s1,s2));
        result.insert(intersection2(s2,s3));
        result.insert(intersection2(s1,s3));

        return set2vector(result);
    }
};
```