```
#include <iostream>
#include <string>
#include <algorithm>
#include <fstream>
#include <vector>
#include <cstring>
using namespace std;

#include "ctimer.h"

class PointerProxy
{
  public:
    PointerProxy(char * s)
        : myPtr(s),
          myLength(strlen(s))
    {

    }

    bool operator < (const PointerProxy& rhs) const{
        int minLength = myLength < rhs.myLength ? myLength : rhs.myLength;
        int result =  strncmp(myPtr, rhs.myPtr, minLength);
        return result == 0 ? myLength < rhs.myLength : (result < 0);
    }

    bool operator == (const PointerProxy& rhs) const{
        int minLength = myLength < rhs.myLength ? myLength : rhs.myLength;
        int result =  strncmp(myPtr, rhs.myPtr, minLength);
        return result == 0 && myLength == rhs.myLength;
    }

    void print(ostream& out) const{
        out << myPtr;
    }

  private:

    char * myPtr;
    int    myLength;
};

ostream& operator << (ostream& out, const PointerProxy& pp)
{
    pp.print(out);
    return out;
}

template <class Type>
double timesort(vector<Type>& v)
{
    CTimer timer;
    timer.Start();
    sort(v.begin(), v.end());
    timer.Stop();
    return timer.ElapsedTime();
}

template <class Type>
bool issorted(vector<Type>& v)
{
    for(int k=1; k < v.size(); k++){
        if (! (v[k-1] <= v[k])) return false;
    }
    return true;
}

int main(int argc, char * argv[])
{
```



```
    string filename;

    if (argc > 1){
        filename = argv[1];
    }
    else {
        cout << "filename: ";
        cin >> filename;
    }
    string word;
    vector<string> vs;
    vector<const char *> vsp;

    ifstream input(filename.c_str());
    while (input >> word){
        vs.push_back(word);
        vsp.push_back(word.c_str());
    }

    double stime = timesort(vs);
    if (! issorted(vs)){
        cout << "string not sorted" << endl;
    }
    if (issorted(vsp)){
        cout << "surprise, proxy sorted" << endl;
    }

    double ptime = timesort(vsp);
    if (! issorted(vsp)){
        cout << "proxy not sorted" << endl;
    }

    cout << "string time = " << stime << endl;
    cout << "proxy time = " << ptime << endl;
}
```