```
#include <iostream>
#include <algorithm>
using namespace std;

#include "tvector.h"
#include "randgen.h"
#include "ctimer.h"

class Finder
{
  public:
    virtual ~Finder() { }
    virtual int find(int kindex, tvector<int>& v){
        sort(v.begin(), v.end());
        return v[kindex];
    }
};

class FastFinder : public Finder
{
   public:
     virtual ~FastFinder() { }
     virtual int find(int kindex, tvector<int>& v){
          return findHelper(kindex,v, 0, v.size()-1);
     }

   protected:
     int findHelper(int kindex, tvector<int>& v, int first, int last);

     void swap(tvector<int>& v, int a, int b)
     {
         int temp = v[a];
         v[a] = v[b];
         v[b] = temp;
     }
};

int FastFinder::findHelper(int kindex, tvector<int>& v, int first, int last)
{
    int lastIndex = first;
    int pivot = v[first];
    for(int k=first+1; k <= last; k++){
        if (v[k] <= pivot){
            lastIndex++;
            swap(v,lastIndex,k);
        }
    }
    swap(v,lastIndex,first);

    if (lastIndex == kindex) return v[lastIndex];
    if (kindex < lastIndex ) return findHelper(kindex,v,first,lastIndex);
    return findHelper(kindex, v,lastIndex+1,last);
}


double stress(int trials, tvector<int> v, Finder * finder)
{
    CTimer timer;
    tvector<int> copy;
    RandGen gen;
    timer.Start();
    for(int k = 0; k < trials; k++){
        copy = v;
        int index = gen.RandInt(0,v.size()-1);

        int f = finder->find(index,copy);

        if (f != index){
            cout << "fail on " << index << " with " << f << endl;
```



```
        }
    }
    timer.Stop();
    return timer.ElapsedTime();
}

int main()
{
    tvector<int> v;
    for(int k=0; k < 500000; k++){
        v.push_back(k);
    }
    RandGen gen;
    for(int k=0; k < v.size()-1; k++){
        int n = gen.RandInt(k+1,v.size()-1);
        int temp = v[n];
        v[n] = v[k];
        v[k] = temp;
    }
    Finder * easyfinder = new Finder();
    Finder * fastfinder = new FastFinder();

    double etime = stress(10,v, easyfinder);
    double ftime = stress(10,v, fastfinder);
    cout << "easy time = " << etime << endl;
    cout << "fast time = " << ftime << endl;
}
```