```
Script started on Sun 14 Jul 2013 12:22:49 AM CDT
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ pwd
/home/jakob/cplusplus
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat timet
ests.info
Name: Jakob Hansen
Class: CSC216
Lab: Date Class, assignment 6.15

Levels attempted:

3


Program Description:

A program to plot the runtime of different algorithms across a range of input
sizes (represented as dots) as well as simple polynomial reference functions
(represented as a solid line). Menu choices are the 7 fragments from the book
plus insertionsort and quicksort, which sort vectors of random numbers.
\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat quick
sort.h
#ifndef QUICKSORT_H
#define QUICKSORT_H

#include<iostream>
#include<vector>
#include<string>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;

const short CUTOFF = 5;

bool doubleEql(const double lhs, const double rhs)
{
        return fabs(lhs-rhs) < 1e-6;
}


inline vector<short> randShort(vector<short>::size_type n)
{
        vector<short> vec;
        vector<short>::size_type i;
        for (i=0; i < n; i++)
        {
                vec.push_back(static_cast<short>(rand()));
        }
        return vec;
}

template <class Comparable>

inline void swap2(Comparable & a, Comparable & b)
{
        Comparable c = a;
        a = b;
        b = c;
        return;
}

template <class Comparable, typename compSizeT>

void insertionSort(vector<Comparable> & a, compSizeT low, compSizeT high)
{
        for (compSizeT p = low; p <= high; p++)
        {
                Comparable tmp = a[p];
                compSizeT j;

                for (j = p; j > low && tmp < a[j-1]; j--)
                {
                        a[j] = a[j-1];
                }
                a[j] = tmp;
        }
        return;
}


template <class Comparable, typename compSizeT>

void quicksort(vector<Comparable> & a, compSizeT low, compSizeT high)
{
        if (low + CUTOFF > high)
        {
                insertionSort(a, low, high);
        }
        else
        {
                compSizeT middle = (low+high)/2;

                if (a[middle] < a[low])
                {
                        swap2(a[low], a[middle]);
                }
                if (a[high] < a[low])
                {
                        swap2(a[low], a[high]);
                }
                if (a[high] < a[middle])
                {
                        swap2(a[middle], a[high]);
                }

                Comparable pivot = a[middle];

                swap2(a[middle], a[high-1]);

                compSizeT i, j;

                for (i = low-1, j = high-1; ; )
                {
                        while(a[++i] < pivot) { }  // find low

                        while (pivot < a[--j]) { } // find high

                        if (i < j)
                        {
                                swap2(a[i], a[j]);
                        }
                        else
                        {
                                break;
                        }
                }
                swap2(a[i], a[high-1]);

                quicksort(a, low, i-1);
                quicksort(a, i+1, high);
        }
        return;
}

template <class Comparable>
void quicksort(vector<Comparable> & a)
{
```

```
        typename vector<Comparable>::size_type b=0;

        quicksort(a, b, a.size()-1);
        return;
}

#endif

\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ cat timet
ests.cpp
#include <sys/time.h>
#include <iostream>
#include <cmath>
#include <climits>
#include "ccc_win.h"
#include <string>
#include <sstream>
#include <vector>
#include "quicksort.h"

using namespace std;

// runs an algorithm n times and samples runtimes at given intervals
template <typename argT, typename intT, typename funcT>
vector<double> timevec(argT n, intT rate, funcT & algo)
{
    vector<double> times;
    double diff;
    timeval start, end;
    for (argT i=0; i<n; i+=rate)
    {
        gettimeofday(&start,NULL);
        algo(i);
        gettimeofday(&end,NULL);
    diff = end.tv_sec+end.tv_usec*1e-6 -
                (start.tv_sec+start.tv_usec*1e-6);
    times.push_back(diff*1e6);
    }
    return times;
}




void drawAxes(double scale=15.0);

// plots values in a vector in a hopefully pretty way
void plotvector(vector<double> vec, long n, double ymax,
                      bool drawline=false, double scale=15.0);

/* prompts for coefficients of 4 terms of a simple polynomial starting with
constant term and plots it, also in a hopefully pretty way */
void plotPolynomial(double xmax, double ymax, double win_size=15.0);

bool emptyabove(vector<double> vec, vector<double>::size_type start);

// dummy algorithms from the book
void function1(long n);
void function2(long n);
void function3(long n);
void function4(long n);
void function5(long n);
void function6(long n);
void function7(long n);
vector<double> timeins(vector<short>::size_type n,
                           vector<short>::size_type rate)
{
    vector<double> times;
    vector<short> v;
    double diff;
    timeval start, end;
    for (vector<short>::size_type i=5; i<n; i+=rate)
    {
        v = randShort(i);
        gettimeofday(&start,NULL);
        insertionSort(v, static_cast<vector<short>::size_type>(0), v.size()-1);
        gettimeofday(&end,NULL);
    diff = end.tv_sec+end.tv_usec*1e-6 -
                (start.tv_sec+start.tv_usec*1e-6);
    times.push_back(diff*1e6);
    }
    return times;
}
vector<double> timequick(vector<short>::size_type n,
                             vector<short>::size_type rate)
{
    vector<double> times;
    vector<short> v;
    double diff;
    timeval start, end;
    for (vector<short>::size_type i=5; i<n; i+=rate)
    {
        v = randShort(i);
        gettimeofday(&start,NULL);
        quicksort(v);
        gettimeofday(&end,NULL);
    diff = end.tv_sec+end.tv_usec*1e-6 -
                (start.tv_sec+start.tv_usec*1e-6);
    times.push_back(diff*1e6);
    }
    return times;
}
int ccc_win_main(void)
{
        typedef void(*pfunc)(long);
        pfunc pfuncs[9] = { function1,
                            function2,
                            function3,
                            function4,
                            function5,
                            function6,
                            function7 };
    ostringstream prompt;

    unsigned long max;
    vector<double> times;
    string in;
    size_t func;

    do {
    // Get N value and take samples of running times for increasing values of n
    func = cwin.get_int("Which function (1 through 9): ");
    prompt << "Enter N value for function " << func << ": ";
    max = cwin.get_int(prompt.str());
    if (func == 8)
    {
        times = timeins(max, max/100);
    }
    else if (func == 9)
    {
        times = timequick(max, max/100);
    }
    else
    {
        times = timevec(max, max/100, pfuncs[func-1]);
    }

    /* superimpose plots of polynomials over the runtimes till you find one that
```

```
    matches nicely! */
                do
                {
                        drawAxes();
                        plotvector(times, max, times[times.size()-1]);
                        plotPolynomial(max, times[times.size()-1]);
                        in = cwin.get_string("Would you like to try again? ");
                        cwin.clear();
                }while(in == "Y");
                times.clear();

    in = cwin.get_string("Would you like to continue? ");
    prompt.str("");
    } while(in == "Y");

    return 0;
}


void plotvector(vector<double> vec, long n, double ymax,
                                bool drawline, double scale)
{
    double xmax = static_cast<double>(vec.size()-1);
    ostringstream label;
    label << "N = " << n;
    cwin << Message(Point(-2, scale), ymax);
    cwin << Message(Point(scale, -1), n);
    cwin << Message(Point(scale/4, scale+3), label.str());
    for (short i=0; i< xmax; i++)
    {
        if (drawline)
        {
            cwin << Line(Point(i/(xmax/scale), vec[i]/(ymax/scale)),
                         Point((i+1)/(xmax/scale), vec[i+1]/(ymax/scale)));
        }
        else
        {
            cwin << Point(i/(xmax/scale), vec[i]/(ymax/scale));
        }
    }
    return;
}

void plotPolynomial(double xmax, double ymax, double win_size)
{
        ostringstream label;
    vector<double> terms;
    vector<double>::size_type term_index;
    double x_value = 0,
           y_value = 0,
           next_x = 0,
           next_y = 0,
           x_ratio = win_size/xmax,
           y_ratio = win_size/ymax,
           coef;

    coef = cwin.get_double("Constant term: ");
    terms.push_back(coef);
    coef = cwin.get_double("Linear coefficient: ");
    terms.push_back(coef);
    coef = cwin.get_double("Quadratic coefficient: ");
    terms.push_back(coef);
    coef = cwin.get_double("Cubic coefficient: ");
    terms.push_back(coef);

    // Print a nice label for the function

    label << "f(n) = ";
    for (term_index = terms.size()-1; term_index > 0; term_index--)
    {
        if (!doubleEql(terms[term_index], 0.0) &&
                  !emptyabove(terms, term_index))
        {
                label << " + ";
        }
        if (!doubleEql(terms[term_index], 0.0)  && term_index > 1)
        {
                label << terms[term_index] << "n^" << term_index;
        }
        else if (term_index == 1 && !doubleEql(terms[term_index], 0.0) )
        {
                label << terms[term_index] << 'n';
        }

    }
    if (!doubleEql(terms[0], 0.0)  && !emptyabove(terms, 0))
    {
        label << " + " << terms[0];
    }

    cwin << Message(Point(win_size/4, win_size+2), label.str());

    while (x_value < xmax && y_value < ymax)
    {
        y_value = 0;
        next_y = 0;
        next_x = 0;

        /* Vector indeces correspond to the exponents of the polynomial terms.
        For each member of the vector, raise x to the power of the index,
        multiply it by the coefficent that lives at that index. Take the sum of
        all the terms to get the f(x) value for the first of two points between
        which to draw a short line. */

        for (term_index = 1; term_index < terms.size(); term_index++)
        {
            y_value += terms[term_index]*pow(x_value, term_index) + terms[0];
        }

        // increment our x value by 1/100th of the max x value.
        next_x = x_value + xmax/100;

        // calculate f(x) same as above for the second point
        for (term_index = 1; term_index < terms.size(); term_index++)
        {
            next_y += terms[term_index]*pow(next_x, term_index) + terms[0];
        }

        // draw the line
        cwin << Line(Point(x_value*x_ratio, y_value*y_ratio),
                     Point(next_x*x_ratio, next_y*y_ratio));

        // set beginning x value to the one we just finished
        x_value = next_x;
    }
    return;
}

void drawAxes(double scale)
{
    cwin << Line(Point(0,0), Point(scale,0));
    cwin << Line(Point(0,0), Point(0, scale));
    return;
}

void function1(long n)
{
    long sum;
```

```
    for (long i=0; i<n; i++)
    {
        sum++;
    }
    return;
}
void function2(long n)
{
    long sum;
    for (long i = 0; i<n; i+=2)
    {
        sum++;
    }
    return;
}
void function3(long n)
{
    long sum;
    for (long i = 0; i<n; i++)
    {
        for (long j = 0; j<n; j++)
        {
            sum++;
        }
    }
    return;
}

void function4(long n)
{
    long sum;
    for (long i = 0; i<n; i++)
    {
        sum++;
    }
    for (long j = 0; j<n; j++)
    {
        sum++;
    }
    return;
}
void function5(long n)
{
    long sum;
    for (long i = 0; i<n; i++)
    {
        for (long j = 0; j<n*n; j++)
        {
            sum++;
        }
    }
    return;
}

void function6(long n)
{
    long sum;
    for (long i = 0; i < n; i++)
    {
        for (long j = 0; j < i; j++)
        {
            sum++;
        }
    }
    return;
}

void function7(long n)
{
    long sum;
    for (long i = 0; i < n; i++)
    {
        for (long j = 0; j < n*n; j++)
        {
            for (long k = 0; k < j; k++)
            {
                sum++;
            }
        }
    }
    return;
}
bool emptyabove(vector<double> vec, vector<double>::size_type start)
{
        vector<double>::size_type i;
        bool returnval = true;
        for(i=start+1; i < vec.size(); i++)
        {
                if (!doubleEql(vec[i], 0.0))
                {
                        returnval = false;
                }
        }
        return returnval;
}


\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ CPP timet
ests ccc_x11 ccc_shap quicksort
ccc_shap.cpp...
ccc_x11.cpp...
timetests.cpp..*
In file included from ccc_x11.cpp:31:0:
In file included from ccc_win.h:36:0,
                 from timetests.cpp:5:


\033]0;jakob@jakob-MT6017: ~/cplusplus\007jakob@jakob-MT6017:~/cplusplus$ exit
exit

Script done on Sun 14 Jul 2013 12:23:38 AM CDT
```