#include <iostream>

#include <string>

#include <cstring>

#include <set>

#include <map>

#include <iterator>

#include <vector>

#include <fstream>

#include <ctime>

#include <list>

#include <queue>

#include <memory>

using namespace std;

// practice 1

bool Palindrome(string input)

{

    string::iterator forware = input.begin();

    string::reverse\_iterator backware = input.rbegin();

    int input\_length = input.length();

    for (int i = 0; i < input\_length / 2; i++)

    {

        if (\*forware == \*backware)

        {

            forware++;

            backware++;

        }

        else

            return false;

    }

    return true;

}

void p16\_1(void)

{

    string input;

    cout << "Enter the string: ";

    getline(cin, input);

    if (Palindrome(input))

    {

        cout << input << " is a palindrome." << endl;

    }

    else

    {

        cout << input << " is not a palindrome." << endl;

    }

}

// practice 2

void p16\_2(void)

{

    string input;

    string temp;

    cout << "Enter string: ";

    getline(cin, input);

    temp = input;

    for (string::iterator it = input.begin(); it != input.end();)

    {

        if (!isalpha(\*it))

        {

            it = input.erase(it);

            continue;

        }

        else

        {

            \*it = tolower(\*it);

            it++;

        }

    }

    if (Palindrome(input))

    {

        cout << input << " is a palindrome." << endl;

    }

    else

    {

        cout << input << " is not a palindrome." << endl;

    }

}

// practice 3

const string FileName = "word.txt";

void p16\_3(void)

{

    vector<string> wordlist;

    ifstream in;

    in.open(FileName.c\_str());

    string inword;

    while (in >> inword)

    {

        wordlist.push\_back(inword);

    }

    srand(time(0));

    char play;

    cout << "Will you play a word game? <y/n> ";

    cin >> play;

    play = tolower(play);

    while (play == 'y')

    {

        string target = wordlist[rand() % wordlist.size()];

        int length = target.length();

        string attemp(length, '-');

        string badchars;

        int guesses = 6;

        cout << "Guess my scret word. It has " << length

            << " letters, and you guess\n"

            << "one letter at a time. You get " << guesses

            << " wrong guesses.\n";

        cout << "Your word: " << attemp << endl;

        while (guesses > 0 && attemp != target)

        {

            char letter;

            cout << "Guess a letter: ";

            cin >> letter;

            if (badchars.find(letter) != string::npos

                || attemp.find(letter) != string::npos)

            {

                cout << "You already gueesed that. Try again.\n";

                continue;

            }

            int loc = target.find(letter);

            if (loc == string::npos)

            {

                cout << "Oh, bad guess!\n";

                --guesses;

                badchars += letter;

            }

            else

            {

                cout << "Good guess!\n";

                attemp[loc] = letter;

                loc = target.find(letter, loc + 1);

                while (loc != string::npos)

                {

                    attemp[loc] = letter;

                    loc = target.find(letter, loc + 1);

                }

            }

            cout << "You word: " << attemp << endl;

            if (attemp != target)

            {

                if (badchars.length() > 0)

                    cout << "Bad choices: " << badchars << endl;

                cout << guesses << " bad guesses left\n";

            }

        }

        if (guesses > 0)

        {

            cout << "That's right!\n";

        }

        else

        {

            cout << "Sorry, the word is " << target << ".\n";

        }

        cout << "Will you play anther? <y/n> ";

        cin >> play;

        play = tolower(play);

    }

    cout << "Bye\n";

    return;

}

// practice 4

int reduce(long ar[], int n)

{

    list<long> lar(ar, ar + n);

    lar.sort();

    lar.unique();

    int lar\_length = 0;

    for (list<long>::iterator it = lar.begin(); it != lar.end(); it++)

    {

        \*(ar + lar\_length) = \*it;

        lar\_length++;

    }

    for (int i = lar\_length; i < n; i++)

    {

        \*(ar + lar\_length) = 0;

    }

    return lar\_length;

}

void p16\_4(void)

{

    long ar[] = { 12, 2, 13, 12, 2, 55, 32, 44, 32, 100, 32, 12 };

    int ar\_length = 12;

    int reduce\_length = 0;

    cout << "original array length: " << ar\_length << endl;

    cout << "original array: ";

    for (int i = 0; i < ar\_length; i++)

    {

        cout << ar[i] << " " ;

    }

    cout << endl;

    reduce\_length = reduce(ar, ar\_length);

    cout << "After reduce, array length: " << reduce\_length << endl;

    cout << "After reduce, array: ";

    for (int i = 0; i < reduce\_length; i++)

    {

        cout << ar[i] << " " ;

    }

    cout << endl;

}

// practice 5

template <class T>

int Treduce(T ar[], int n)

{

    list<T> lar(ar, ar + n);

    lar.sort();

    lar.unique();

    int lar\_length = 0;

    for (list<T>::iterator it = lar.begin(); it != lar.end(); it++)

    {

        \*(ar + lar\_length) = \*it;

        lar\_length++;

    }

    return lar\_length;

}

void p16\_5(void)

{

    // long

    long ar[] = { 12, 2, 13, 12, 2, 55, 32, 44, 32, 100, 32, 12 };

    int ar\_length = 12;

    int reduce\_length = 0;

    cout << "original array length: " << ar\_length << endl;

    cout << "original array: ";

    for (int i = 0; i < ar\_length; i++)

    {

        cout << ar[i] << " ";

    }

    cout << endl;

    reduce\_length = Treduce(ar, ar\_length);

    cout << "After reduce, array length: " << reduce\_length << endl;

    cout << "After reduce, array: ";

    for (int i = 0; i < reduce\_length; i++)

    {

        cout << ar[i] << " ";

    }

    cout << endl;

    cout << endl;

    // string

    string sar[] = { "hello", "jimmy", "array", "hehe", "jimmy", "hehe" };

    int sar\_length = 6;

    cout << "original string array length: " << sar\_length << endl;

    cout << "original string array: ";

    for (int i = 0; i < sar\_length; i++)

    {

        cout << sar[i] << " ";

    }

    cout << endl;

    reduce\_length = Treduce(sar, sar\_length);

    cout << "After reduce, array length: " << reduce\_length << endl;

    cout << "After reduce, array: ";

    for (int i = 0; i < reduce\_length; i++)

    {

        cout << sar[i] << " ";

    }

    cout << endl;

}

// practice 6

class Customer {

private:

    long arrive;

    int processtime;

public:

    Customer() { arrive = processtime = 0; }

    void set(long when) {

        processtime = rand() % 3 + 1;

        arrive = when;

    }

    long when() const { return arrive; }

    int ptime() const { return processtime; }

};

const int MIN\_PER\_HR = 60;

bool newcustomer(double x)

{

    return (rand() \* x / RAND\_MAX < 1);

}

void p16\_6(void)

{

    srand(time(0));

    cout << "Case study: Band of Heather Automatic Teller\n";

    cout << "Enter maximum size of queue: ";

    int qs;

    cin >> qs;

    queue<Customer> line;

    cout << "Enter the number of simulation hours: ";

    int hours;

    cin >> hours;

    long cyclelimit = MIN\_PER\_HR \* hours;

    cout << "Enter the average number of customer per hours: ";

    double perhour;

    cin >> perhour;

    double min\_per\_cust;

    min\_per\_cust = MIN\_PER\_HR / perhour;

    Customer temp;

    long turnaways = 0;

    long customers = 0;

    long served = 0;

    long sum\_line = 0;

    int wait\_time = 0;

    long line\_wait = 0;

    for (int cycle = 0; cycle < cyclelimit; cycle++)

    {

        if (newcustomer(min\_per\_cust))

        {

            if (qs == line.size())

                turnaways++;

            else

            {

                customers++;

                temp.set(cycle);

                line.push(temp);

            }

        }

        if (wait\_time <= 0 && !line.empty())

        {

            temp = line.front();

            wait\_time = temp.ptime();

            line\_wait += cycle - temp.when();

            served++;

            line.pop();

        }

        if (wait\_time > 0)

        {

            wait\_time--;

        }

        sum\_line += line.size();

    }

    if (customers > 0)

    {

        cout << "customers accepted: " << customers << endl;

        cout << "  customers served: " << served << endl;

        cout << "         turnaways: " << turnaways << endl;

        cout << "average queue size: ";

        cout.precision(2);

        cout.setf(ios\_base::fixed, ios\_base::floatfield);

        cout << (double)sum\_line / cyclelimit << endl;

        cout << " average wait time: "

            << (double)line\_wait / served << " minutes\n";

    }

    else

        cout << "No customers!\n";

    cout << "Done!\n";

    return;

}

// practice 7

vector<int> Lotto(int total, int choice)

{

    vector<int> ar(total);

    for (int i = 0; i < total; i++)

    {

        ar[i] = i + 1;

    }

    random\_shuffle(ar.begin(), ar.end());

    vector<int> car(choice);

    for (int i = 0; i < choice; i++)

    {

        car[i] = ar[i];

    }

    sort(car.begin(), car.end());

    return car;

}

void p16\_7(void)

{

    srand(time(0));

    vector<int> winners;

    winners = Lotto(51, 6);

    cout << "Winners' number: ";

    for (size\_t i = 0; i < 6; i++)

    {

        cout << winners[i] << " ";

    }

    cout << endl;

    return;

}

// practice 8

void p16\_8(void)

{

    set<string> Mat\_friend;

    set<string> Pat\_friend;

    set<string> union\_friend;

    string input;

    cout << "Mat friend: ";

    while (cin >> input)

    {

        Mat\_friend.insert(input);

        if (cin.get() == '\n')

        {

            break;

        }

    }

    copy(Mat\_friend.begin(), Mat\_friend.end(), ostream\_iterator<string, char>(cout, " "));

    cout << endl;

    cout << "Pat friend: ";

    while (cin >> input)

    {

        Pat\_friend.insert(input);

        Mat\_friend.insert(input);

        if (cin.get() == '\n')

        {

            break;

        }

    }

    copy(Pat\_friend.begin(), Pat\_friend.end(), ostream\_iterator<string, char>(cout, " "));

    cout << endl;

    set\_union(Mat\_friend.begin(), Mat\_friend.end(), Pat\_friend.begin(), Pat\_friend.end(), insert\_iterator<set<string> >(union\_friend, union\_friend.begin()));

    cout << "All friends: ";

    copy(union\_friend.begin(), union\_friend.end(), ostream\_iterator<string, char>(cout, " "));

    cout << endl;

}

// practice 9

const long VEC\_SIZE = 1000000;  // 这个量级i5-3320M大概用了一分多钟

void p16\_9(void)

{

    cout << "-----start-----" << endl;

    srand(time(0));

    vector<int> vi0(VEC\_SIZE);

    for (long i = 0; i < VEC\_SIZE; i++)

    {

        vi0[i] = rand() % 10000 + 1;

    }

    vector<int> vi(vi0);

    list<int> li(vi0.begin(), vi0.end());

    clock\_t vector\_start = clock();

    sort(vi.begin(), vi.end());

    clock\_t vector\_end = clock();

    clock\_t list\_start = clock();

    li.sort();

    clock\_t list\_end = clock();

    copy(vi0.begin(), vi0.end(), li.begin());

    clock\_t copy\_sort\_time\_start = clock();

    copy(li.begin(), li.end(), vi.begin());

    sort(vi.begin(), vi.end());

    copy(vi.begin(), vi.end(), li.begin());

    clock\_t copy\_sort\_time\_end = clock();

    cout << "test vector size: " << VEC\_SIZE << endl;

    cout << "vector sort time: " << (double)(vector\_end - vector\_start) / CLOCKS\_PER\_SEC << endl;

    cout << "list sort time: " << (double)(list\_end - list\_start) / CLOCKS\_PER\_SEC << endl;

    cout << "list copy sort time: " << (double)(copy\_sort\_time\_end - copy\_sort\_time\_start) / CLOCKS\_PER\_SEC << endl;

    cout << "-----end-----" << endl;

}

// practice 10

struct Review

{

    string title;

    int rating;

    double price;

};

bool FillRevice(Review & rr)

{

    cout << "Enter book title (quit to quit): ";

    getline(cin, rr.title);

    if (rr.title == "quit")

    {

        return false;

    }

    cout << "Enter book rating: ";

    cin >> rr.rating;

    if (!cin)

    {

        return false;

    }

    cout << "Enter book price: ";

    cin >> rr.price;

    if (!cin)

    {

        return false;

    }

    while (cin.get() != '\n')

        continue;

    return true;

}

void ShowReview(const shared\_ptr<Review> & rr)

{

    cout << rr->rating << "\t" << rr->title << "\t" << rr->price << endl;

}

bool operator<(const shared\_ptr<Review> & r1, const shared\_ptr<Review> & r2)

{

    if (r1->title < r2->title)

    {

        return true;

    }

    else if (r1->title == r2->title && r1->rating < r2->rating)

        return true;

    else

        return false;

}

bool worseThan(const shared\_ptr<Review> & r1, const shared\_ptr<Review> & r2)

{

    if (r1->rating < r2->rating)

        return true;

    else

        return false;

}

bool betterThan(const shared\_ptr<Review> & r1, const shared\_ptr<Review> & r2)

{

    if (r1->rating > r2->rating)

        return true;

    else

        return false;

}

bool expensiveThan(const shared\_ptr<Review> & r1, const shared\_ptr<Review> & r2)

{

    if (r1->price > r2->price)

        return true;

    else

        return false;

}

bool cheaperThan(const shared\_ptr<Review> & r1, const shared\_ptr<Review> & r2)

{

    if (r1->price < r2->price)

        return true;

    else

        return false;

}

void p16\_10(void)

{

    vector<shared\_ptr<Review> > books;

    Review temp;

    while (FillRevice(temp))

    {

        shared\_ptr<Review> in(new Review(temp));

        books.push\_back(in);

    }

    cout << "Enter your choice: " << endl;

    cout << "0：按原始顺序显示， 1：按字母表顺序显示， 2：按评级升序显示" << endl;

    cout << "3：按评级降序显示， 4：按价格升序显示  ， 5：按价格降序显示" << endl;

    cout << "6：退出" << endl;

    int choice = 0;

    vector<shared\_ptr<Review> > original\_books(books.begin(), books.end());

    while (cin >> choice && choice != 6)

    {

        switch (choice)

        {

        case 0:

            copy(original\_books.begin(), original\_books.end(), books.begin());

            break;

        case 1:

            sort(books.begin(), books.end());

            break;

        case 2:

            sort(books.begin(), books.end(), worseThan);

            break;

        case 3:

            sort(books.begin(), books.end(), betterThan);

            break;

        case 4:

            sort(books.begin(), books.end(), cheaperThan);

            break;

        case 5:

            sort(books.begin(), books.end(), expensiveThan);

            break;

        default:

            break;

        }

        for\_each(books.begin(), books.end(), ShowReview);

    }

    return;

}

int main(int argc, char \*\* argv)

{

    p16\_10();

    while (cin.get());

    return 0;

}