1、

//Reverse.cpp

#include <iostream>

#include <string>

using namespace std;

bool isReverse(const string &s1, const string &s2);

int main()

{

string words;

cout << "Enter a string (quit to quit): \n";

while (cin >> words&&words != "quit")

{

string rwords(words.rbegin(), words.rend());

cout << "words: " << words << endl;

if (isReverse(rwords, words))

cout << words << " is reverse words.\n";

else

cout << words << " is not reverse words.\n";

cout << "Enter a next words (quit to quit): \n";

}

system("pause");

return 0;

}

bool isReverse(const string &s1, const string &s2)

{

if (s1.compare(s2))

return false;

else

return true;

}

2、

//Reverse.cpp

#include <iostream>

#include <string>

#include <cctype>

using namespace std;

bool isReverse(const string &s1, const string &s2);

void ToLower(string &s1);

int main()

{

string words;

cout << "Enter a string (quit to quit): \n";

getline(cin, words);

while (words != "quit")

{

ToLower(words);

string rwords(words.rbegin(), words.rend());

cout << "words: " << words << endl;

if (isReverse(rwords, words))

cout << words << " is reverse words.\n";

else

cout << words << " is not reverse words.\n";

cout << "Enter a next words (quit to quit): \n";

getline(cin, words);

}

system("pause");

return 0;

}

bool isReverse(const string &s1, const string &s2)

{

if (s1.compare(s2))

return false;

else

return true;

}

void ToLower(string &s1)

{

unsigned int i = 0;

while (i < s1.length())

{

if (isalpha(s1[i]))

{

s1[i] = tolower(s1[i]);

i++;

}

else

s1.replace(i, s1.size(), s1, i + 1, s1.size());

}

}

3、

//hangman.cpp

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <cctype>

#include <fstream>

#include <vector>

using namespace std;

vector<string> wordlist;

void FillWord();

int main()

{

srand(time(0));

char play;

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

cin >> play;

play = tolower(play);

FillWord();

while (play == 'y')

{

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

int length = target.length();

string attempt(length, '-');

string badchars;

int guesses = 6;

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

<< " letters, and you guess\n"

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

<< " wrong guesses.\n";

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

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

{

char letter;

cout << "Guess a letter: ";

cin >> letter;

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

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

{

cout << "You already guessed 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";

attempt[loc] = letter;

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

while (loc != string::npos)

{

attempt[loc] = letter;

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

}

}

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

if (attempt != 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 another? <y/n> ";

cin >> play;

play = tolower(play);

}

cout << "Bye\n";

system("pause");

return 0;

}

void FillWord()

{

ifstream fin;

string letter;

fin.open("letters.txt");

if (fin.is\_open() == false)

{

cerr << "Can't open file. Bye.\n";

exit(EXIT\_FAILURE);

}

while (fin)

{

fin >> letter;

wordlist.push\_back(letter);

}

fin.close();

}

//letters.txt

apiary beetle cereal danger ensign florid garage health

insult jackal keeper loaner manage nonce onset plaid

quilt remote stolid train useful valid whence xenon yearn zippy

4、

//Reduce.cpp

#include <iostream>

#include <list>

#include <algorithm>

using namespace std;

int reduce(long ar[], int n);

void outlong(int n){ cout << n << " "; }

int main()

{

long a[10] = { 1, 3, 2, 4, 7, 6, 3, 4, 8, 10 };

int nums = reduce(a, 10);

cout << "Numbers after reduce: " << nums << endl;

system("pause");

return 0;

}

int reduce(long ar[], int n)

{

list<long> arr;

arr.insert(arr.begin(), ar, ar + n);

arr.sort();

arr.unique();

for\_each(arr.begin(), arr.end(), outlong);

cout << endl;

return arr.size();

}

5、

//Reduce.cpp

#include <iostream>

#include <list>

#include <algorithm>

#include <string>

using namespace std;

template <class T>

int reduce(T ar[], int n);

template <class T>

void outlong(T n){ cout << n << " "; }

int main()

{

long a[10] = { 1, 3, 2, 4, 7, 6, 3, 4, 8, 10 };

int nums1 = reduce(a, 10);

cout << "Numbers after reduce: " << nums1 << endl;

string b[10] = { "qwe", "qwe", "ret", "tyu", "dfg",

"jkl", "iok", "kjl", "ads", "def" };

int nums2 = reduce(b, 10);

cout << "Numbers after reduce: " << nums2 << endl;

system("pause");

return 0;

}

template <class T>

int reduce(T ar[], int n)

{

list<T> arr;

arr.insert(arr.begin(), ar, ar + n);

arr.sort();

arr.unique();

for\_each(arr.begin(), arr.end(), outlong<T>);

cout << endl;

return arr.size();

}

6、

//bank.cpp

#include <iostream>

#include <cstdlib>

#include <queue>

#include <ctime>

using namespace std;

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; }

};

typedef Customer Item;

const int MIN\_PER\_HR = 60;

bool newcustomer(double x);

int main()

{

srand(time(0));

cout << "Case Study: Bank of Heather Automatic Teller\n";

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

int qs;

cin >> qs;

queue<Item> line;

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

int hours;

cin >> hours;

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

cout << "Enter the average number of customers per hour: ";

double perhour;

cin >> perhour;

double min\_per\_cust;

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

Item 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 (line.size() == qs)

turnaways++;

else

{

customers++;

temp.set(cycle);

line.push(temp);

}

}

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

{

line.pop();

wait\_time = temp.ptime();

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

served++;

}

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";

system("pause");

return 0;

}

bool newcustomer(double x)

{

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

}

7、

//Lotto.cpp

#include <vector>

#include <iostream>

#include <iterator>

#include <algorithm>

#include <ctime>

using namespace std;

vector<int> Lotto(int l1, int l2);

void Show(int n);

int main()

{

vector<int> winners;

winners = Lotto(51, 6);

for\_each(winners.begin(), winners.end(), Show);

system("pause");

return 0;

}

vector<int> Lotto(int l1, int l2)

{

vector<int> nums;

vector<int> nums2;

srand(time(0));

for (int i = 0; i < l1; i++)

nums.push\_back(i);

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

for (int i = 0; i < l2; i++)

nums2.push\_back(nums[rand() % l1 + 1]);

return nums2;

}

void Show(int n)

{

cout << "The winner is " << n << endl;

}

8、

//name.cpp

#include <iostream>

#include <string>

#include <set>

#include <algorithm>

#include <iterator>

int main()

{

using namespace std;

ostream\_itera

tor<string, char> out(cout, " ");

string fname1;

set<string> A;

cout << "Enter Mat's friends(quit to quit): \n";

getline(cin, fname1);

while (fname1 != "quit")

{

A.insert(fname1);

getline(cin, fname1);

}

cout << "Mat's friends: \n";

copy(A.begin(), A.end(), out);

cout << endl;

string fname2;

set<string> B;

cout << "Enter Pat's friends(quit to quit): \n";

getline(cin, fname1);

while (fname1 != "quit")

{

B.insert(fname1);

getline(cin, fname1);

}

cout << "Pat's friends: \n";

copy(B.begin(), B.end(), out);

cout << endl;

cout << "Union of Mat and Pat's friends: ";

set\_union(A.begin(), A.end(), B.begin(), B.end(), out);

cout << endl;

system("pause");

return 0;

}

9、

//sorttime.cpp

#include <iostream>

#include <vector>

#include <list>

#include <ctime>

#include <iterator>

#include <algorithm>

using namespace std;

const int Size = 10000;

int main()

{

clock\_t start, end;

vector<int> vi0(Size);

list<int> li(Size);

srand(time(0));

for (int i = 0; i < Size; i++)

{

vi0[i] = rand() % Size;

}

copy(vi0.begin(), vi0.end(), back\_inserter(li));

vector<int> vi(vi0);

start = clock();

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

end = clock();

cout << "Time of Sort Vector:" << (double)(end - start) / CLOCKS\_PER\_SEC << endl;

start = clock();

li.sort();

end = clock();

cout << "Time of Sort List:" << (double)(end - start) / CLOCKS\_PER\_SEC << endl;

copy(vi0.begin(), vi0.end(), back\_inserter(li));

start = clock();

copy(li.begin(), li.end(), back\_inserter(vi));

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

copy(vi.begin(), vi.end(), back\_inserter(li));

end = clock();

cout << "Time of Sort List2:" << (double)(end - start) / CLOCKS\_PER\_SEC << endl;

system("pause");

return 0;

}

10、

//vect.cpp

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#include <memory>

#include <cstdlib>

using namespace std;

struct Review

{

string title;

int rating;

double price;

};

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

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

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

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

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

bool FillReview(shared\_ptr<Review> &rr);

shared\_ptr<Review> make\_Review();

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

int main()

{

vector<shared\_ptr<Review>> books;

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

while (FillReview(temp))

{

books.push\_back(temp);

temp = make\_Review();

}

if (books.size() > 0)

{

vector<shared\_ptr<Review>> sbook(books);

cout << "Thank you. You entered the following:\n"

<< books.size() << " ratings:\n" << "Rating\tBook\tPrice\n";

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

char ch;

cout << "Enter measures of sort:\no to old";

cout << "t to title, r to down rating,\nR to up rating"

"p to down price,P to up price, f(F) to shuffle,\nq to quit:";

cin >> ch;

while (tolower(ch) != 'q')

{

switch (ch)

{

case'o':

cout << "Not Sort:\nRating\tBook\tPrice\n";

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

break;

case't':

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

cout << "Sorted by title:\nRating\tBook\tPrice\n";

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

break;

case'r':

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

cout << "Sorted by down rating:\nRating\tBook\tPrice\n";

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

break;

case'R':

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

cout << "Sorted by up rating:\nRating\tBook\tPrice\n";

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

break;

case'p':

sort(sbook.begin(), sbook.end(), worseThanP);

cout << "Sorted by down price:\nRating\tBook\tPrice\n";

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

break;

case'P':

sort(sbook.begin(), sbook.end(), betterThanP);

cout << "Sorted by up price:\nRating\tBook\tPrice\n";

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

break;

case'F':

case'f':

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

cout << "After shuffling:\nRating\tBook\n";

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

break;

default:

cout << "Error input!Input again!";

break;

}

cout << "Enter the next measures:\no to old";

cout << "t to title, r to down rating,\nR to up rating"

"p to down price,P to up price, q to quit:";

cin >> ch;

}

}

else

cout << "No entries. ";

cout << "Bye.\n";

system("pause");

return 0;

}

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

{

if (r1->price < r2->price)

return true;

else

return false;

}

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

{

if (r1->price > r2->price)

return true;

else

return false;

}

shared\_ptr<Review> make\_Review()

{

return shared\_ptr<Review>(new Review);

}

bool FillReview(shared\_ptr<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;

cout << "Enter book price: ";

cin >> rr->price;

if (!cin)

return false;

while (cin.get() != '\n')

continue;

return true;

}

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

{

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

}