// proj 2.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

//#ifndef BAG1\_H

#define BAG1\_H

#include <cstdlib> // Provides size\_t

#include <iostream> // Provides istream and ostream

#include <cassert> // Provides assert()

#include <iomanip> // Provides setw()

#include <fstream>

using namespace std;

const int BAGS = 3; // The number of bags the program manipulates

// Prototypes for functions in this program:

void menu();

// Postcondition: A menu of commands and instructions for their use has been written to cout.

class Bag

{

public:

// TYPEDEF and MEMBER CONSTANT

static const size\_t CAPACITY = 100; // Maximum size of any Bag

typedef int Item; // What can go in a Bag

// CONSTRUCTOR

Bag() { used = 0; } // Inline

// MODIFICATION MEMBER FUNCTIONS

void make\_empty() { used = 0; } // Inline

void insert(const Item& entry);

void remove(const Item& target);

void operator += (const Bag& addend);

// CONSTANT MEMBER FUNCTIONS

size\_t size() const { return used; } // Inline

size\_t occurrences(const Item& target) const;

// FRIEND FUNCTION

friend std::ostream& operator << (std::ostream& out\_s, const Bag& b);

private:

Item data[CAPACITY]; // An array of Items

size\_t used; // How many hold Bag elements

};

// NONMEMBER FUNCTION for this class

Bag operator + (const Bag& b1, const Bag& b2);

//struct Nodes

//{

//

//};

// The default constructor is an inline function.

void Bag::insert(const Item& entry)

// Uses cassert.

{

assert(size() < CAPACITY);

data[used] = entry;

++used;

}

// make\_empty() is an inline function.

void Bag::remove(const Item& target)

// Uses cstdlib.

{

size\_t index; // Where target first appears in data[]

for (index = 0; (index < used) && (data[index] != target); index++)

// Empty loop body

;

if (index == used) // target is not in the bag, so return.

return;

// Remove target from the bag.

used--;

data[index] = data[used];

}

void Bag::operator += (const Bag& addend)

// Uses cassert and cstdlib.

{

size\_t i, addend\_size;

assert(size() + addend.size() <= CAPACITY);

addend\_size = addend.size();

for (i = 0; i<addend\_size; ++i)

{

data[used] = addend.data[i];

++used;

}

}

// size() is an inline function.

size\_t Bag::occurrences(const Item& target) const

// Uses cstdlib.

{

size\_t answer, i;

answer = 0;

for (i = 0; i<used; ++i)

if (data[i] == target)

++answer;

return answer;

}

ostream& operator << (ostream& out\_s, const Bag& b)

// Uses iomanip.

{

out\_s << "{ ";

if (b.used > 1)

for (int i = 0; i<b.used - 1; ++i)

out\_s << setw(1) << b.data[i] << ", ";

if (b.used > 0)

out\_s << setw(1) << b.data[b.used - 1] << ' ';

out\_s << "}";

return out\_s;

}

Bag operator + (const Bag& b1, const Bag& b2)

// Uses cassert.

{

Bag answer;

assert(b1.size() + b2.size() <= Bag::CAPACITY);

answer += b1;

answer += b2;

return answer;

}

int main()

{

char command; // Each command letter

int v; // Each value in a command

int n, n1, n2, n3; // Bag indexes in commands

Bag b[BAGS]; // An array of bags

menu(); // Show the menu.

do

{

cout << "--> "; // Issue a prompt.

cin >> command; // Read a command letter.

switch (command) // Carry out the command.

{

case 'e': cin >> n;

b[n].make\_empty();

break;

case 'i': cin >> v >> n;

b[n].insert(v);

break;

case 'r': cin >> v >> n;

b[n].remove(v);

break;

case 's': cin >> n;

cout << "Bag " << setw(1) << n << " contains "

<< setw(1) << b[n].size() << " items." << endl;

break;

case 'o': cin >> v >> n;

cout << "In bag " << setw(1) << n << ", the value " << setw(1)

<< v << " occurs " << setw(1) << b[n].occurrences(v)

<< " times." << endl;

break;

case 'u': cin >> n1 >> n2 >> n3;

b[n3] = b[n1] + b[n2];

break;

case 'a': cin >> n1 >> n2;

b[n2] += b[n1];

break;

case 'w': cin >> n;

cout << "Bag " << setw(1) << n << ": " << b[n] << endl;

break;

case 'h': menu();

break;

default:; // Null statement for an incorrect command

}

} while (command != 'q');

return EXIT\_SUCCESS;

}

void menu()

{

cout << endl;

cout << "This program responds to commands the user enters to" << endl;

cout << "manipulate " << setw(1) << BAGS << " bags (multisets),"

<< " which are initially" << endl;

cout << "empty. In the following commands, v is any integer, and" << endl;

cout << "n , n1, n2, and n3 are numbers of bags; they may only be" << endl;

cout << "integers from 0 to " << setw(1) << BAGS - 1 << '.' << endl << endl;

cout << " e n -- Re-initialize bag n to be empty." << endl;

cout << " i v n -- Insert the value v into bag n." << endl;

cout << " r v n -- Remove the value v from bag n." << endl;

cout << " s n -- Report the size of bag n." << endl;

cout << " o v n -- Report the number of times the value v" << endl

<< " occurs in bag n." << endl;

cout << " u n1 n2 n3 -- Form the union of bag n1 and bag n2" << endl

<< " put the union in bag n3." << endl;

cout << " a n1 n2 -- Add the contents of bag n1 to bag n2." << endl;

cout << " w n -- Write out the contents of bag n." << endl;

cout << " h -- See this menu." << endl;

cout << " q -- Quit the program." << endl << endl;

}