Створити клас Set, який представляє множину (сукупність елементів без повторень). Реалізувати у цьому класі перевантаження операцій додавання і віднімання.

// ConsoleApplication48.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <string>

using namespace std;

class Set

{

private:

int length, maxLength;

int\* elements;

void set\_maxLength(int value);

int indexOf(int value);

public:

int get\_MaxLength();

int get\_length();

Set(int maxLength,int length,int first,...);

Set(int maxLength);

Set();

Set(Set& obj);

Set& operator+(int value);

Set& operator-(int value);

void add(int value);

bool remove(int value);

bool contains(int value);

string toString();

~Set();

};

//-------------------

int main()

{

/\*

Set s1(20, 4, 11, 22, 33, 22);

printf("Before: %s\n", s1.toString().data());

s1 = s1 + 77; //s1.add(77);

s1 = s1 + 88; //s1.add(88);

s1 = s1 - 11;

printf("After: %s\n", s1.toString().data());

\*/

// З клавіатури вводитяться послідовно ідентифікаційні коди відвідувачів. Визначити скільки різих відвідувачів.

Set\* diffClients = new Set(100);

int idCount;

printf("Count =");

cin >> idCount;

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

{

printf("Id: ");

int id;

cin >> id;

diffClients->add(id); //diffClients = &((\*diffClients) + id);

}

printf("Different clients count=%d\n", diffClients->get\_length());

delete diffClients;

system("pause");

return 0;

}

//=========================================

void Set::set\_maxLength(int value)

{

if (value>0)

{

maxLength = value;

elements = new int[maxLength];

}

else

{

throw "max length must be > 0";

}

}

int Set::indexOf(int value)

{

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

{

if (elements[i]==value)

{

return i;

}

}

return -1;

}

int Set::get\_MaxLength()

{

return maxLength;

}

int Set::get\_length()

{

return length;

}

Set::Set(int maxLength, int length, int first, ...):Set(maxLength)

{

int\* temp = &first;

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

{

add(temp[i]);

}

}

Set::Set(int maxLength)

{

set\_maxLength(maxLength);

}

Set::Set():Set(15)

{

}

Set::Set(Set & obj)

{

set\_maxLength(obj.maxLength);

length = obj.length;

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

{

elements[i] = obj.elements[i];

}

}

//s3 = s1 + 99; //s3=s1.operand+(99)

//s4=s2+76; //s4=s2.operand+(76)

Set& Set::operator+(int value)

{

Set\* temp=new Set(\*this);

temp->add(value);

return \*temp;

}

Set & Set::operator-(int value)

{

Set resultSet(\*this);

resultSet.remove(value);

return resultSet;

}

void Set::add(int value)

{

if (length < maxLength)

{

if (!contains(value))

{

elements[length] = value;

length++;

}

}

else

throw "out of memory";

}

bool Set::remove(int value)

{

int indexOfValue = indexOf(value);

if (indexOfValue!=-1)

{

elements[indexOfValue] = elements[length - 1];

length--;

return true;

}

return false;

}

bool Set::contains(int value)

{

return indexOf(value) != -1;

}

string Set::toString()

{

if (length==0)

{

return "set is empty";

}

string temp = to\_string(elements[0]);

for (int i = 1; i < length; i++)

{

temp += "," + to\_string(elements[i]);

}

return temp;

}

Set::~Set()

{

delete[] elements;

}