#include<iostream>

#include<assert.h>

using namespace std;

template<typename T>

class DynamicArray {

T\* arr;

int size = 0;

public:

DynamicArray():arr(NULL),size(NULL){}

int Size() const {

return size;

}

T& operator[](int index) {

assert(arr != nullptr);

assert(index >=0);

assert(index < size);

return arr[index];

}

DynamicArray(const DynamicArray&other)

{

if (arr != nullptr) {

delete[]arr;

}

if (other.size == 0 || other.arr==nullptr) {

this->arr = nullptr;

this->size = 0;

}

arr = new T[other.size]{};

for (size\_t i = 0; i < other.size; i++)

{

arr[i] = other.arr[i];

}

this->size = other.size;

}

void AddElement(const T& element) {

auto newarray = new T[size + 1]{};

for (size\_t i = 0; i < size; i++)

{

newarray[i] = arr[i];

}

newarray[size] = element;

arr = newarray;

newarray = nullptr;

size++;

}

void RemoveElementByID(const int index)

{

auto newarr = new T[size - 1]{};

for (int x = 0; x < index; x++)

{

newarr[x] = arr[x];

}

for (int y = index + 1; y < size + 1; y++)

{

newarr[y - 1] = arr[y];

}

size--;

arr = newarr;

newarr = nullptr;

}

void Update(const T& user)

{

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

{

if (arr[i] == user)

{

arr[i] = user;

break;

}

}

}

void Print()const {

for (size\_t i = 0; i < size; i++)

{

cout << arr[i] << " ";

} cout << endl;

}

~DynamicArray()

{

delete[]arr;

}

};

class User {

int id;

int age;

char gender;

public:

static int ID;

bool operator==(const User& other) {

return other.id == id;

}

User() = default;

User(int age, char gender) {

id = ID;

ID++;

this->age = age;

this->gender = gender;

}

void SetAge(const int& age)

{

assert(age > 0 && " Age cannot be les than 0");

this->age = age;

}

friend ostream& operator<<(ostream& out, const User& user);

};

ostream& operator<<(ostream& out, const User& user) {

cout << "==============================" << endl;

out << "USER INFO" << endl;

cout << "ID : " << user.id << endl;

out << "Age : " << user.age << endl;

out << "Gender : " << user.gender << endl;

return out;

}

int User::ID = 1;

//class Group {

// DynamicArray<User>users;

//public:

// void AddUser(const User& user) {

// users.AddElement(user);

// }

// void Show() {

// users.Print();

// }

//};

void main() {

/\*DynamicArray<int>arr;

arr.AddElement(1);

arr.AddElement(2);

arr.AddElement(3);

arr.Print();

cout << arr[1] << endl;\*/

User u1(20, 'M');

User u2(32, 'F');

User u3(23, 'M');

User u4(34, 'F');

/\*Group g;

g.AddUser(u1);

g.AddUser(u2);

g.AddUser(u3);

g.Show();\*/

DynamicArray<User>arr;

arr.AddElement(u1);

arr.AddElement(u2);

arr.AddElement(u3);

arr.AddElement(u4);

//arr.Remove(index);

//arr.Update(0, u2);

//arr.GetElementIndex(u1);

arr.Print();

cout << "Index 2 deleted" << endl;

arr.RemoveElementByID(2);

cout << "============================================================" << endl;

arr.Print();

u4.SetAge(99);

arr.Update(u4);

cout << "============================================================" << endl;

arr.Print();

}