#include <iostream>

using namespace std;

class Vector {

public:

Vector(int s = 0);

Vector(const Vector & rhs);

Vector operator =(const Vector & rhs);

~Vector() { // default destructor

}

int & operator[](int index);

int sz();

private:

int size; //store the # of elements used

int \*entries;

};

Vector::Vector(int s) {

size = s; // makes Size = s,

entries = new int[size]; //allocates s space,

for (int index = 0; index < size; index++) { // makes all entries 0

entries[index] = 0;

}

}

Vector::Vector(const Vector& rhs) {

// copy constructor

// makes self a deep copy of rhs

entries = 0;

if (this != &rhs) {

size = rhs.size;

if (size) {

entries = new int[size];

for (int index = 0; index < size; index++) {

entries[index] = rhs.entries[index];

}

}

}

}

Vector Vector::operator=(const Vector& rhs) {

// makes self a deep copy of rhs

entries = 0;

if (this != &rhs) {

size = rhs.size;

if (size) {

entries = new int[size];

for (int index = 0; index < size; index++) {

entries[index] = rhs.entries[index];

}

}

}

return \*this;

}

int & Vector::operator[](int index) {

if (0 <= entries[index] < size) {// if 0 <=pos<size

return entries[index]; // returns entries[index]

}

}

int Vector::sz() {

return size; // returns the # of entries.

}

/\*ostream & operator<<(ostream & out, Vector & rhs);\*/

ostream& operator<<(ostream& out, Vector& rhs) {

for(int i = 0; i < rhs.sz(); i++){

out << rhs[i] << " ";

}

return out;

}

int main() {

Vector test(10);

Vector \*vecPtr;

Vector exam(5);

for (int i = 0; i < test.sz(); i++) {

test[i] = i;

}

cout << "test initally is : ";

cout << test << endl;

cout << "empty Vector of length 5 is : ";

cout << exam << endl;

exam = test;

cout << "after exam = test, exam is : ";

cout << exam << endl;

{

Vector sqrs(10);

cout << "The squares are : ";

for (int i = 0; i < sqrs.sz(); i++)

sqrs[i] = i\*i;

vecPtr = &sqrs;

cout << sqrs << endl;

}

cout << "trying to print an empty vector: ";

cout << \*vecPtr << endl;

for (int i = 0; i < test.sz(); i++)

test[i] = 10 - i;

cout << "testing deep copy :\n";

cout << "test is now : ";

cout << test << endl;

cout << "exam is now : ";

cout << exam << endl;

return 0;

}

Output:

