Demonstrate the use of the Standard Template Library "vector" class.  
// George F. Riley, ECE3090 Georgia Tech, Fall 2005

// A vector is a variable length array. It starts out as "zero" length  
// and grows or shrinks as needed. Further, the vector is a array  
// of any arbitrary type, using the C++ "templates" feature.

#include <iostream>
#include <vector>

class A {
public:
    A(); // Default constructor
    A(int); // Non-Default Constructor
    A(const A&); // A copy constructor is used by the compile whenever
    ~A(); // Destructor
public:
    int x; // Single data member
};

typedef std::vector<A> AVec_t; // Define a type that is vector of A's
typedef std::vector<A*> APVec_t; // Define a type that is vector of A pointers

A::A()
{
    std::cout << "Hello from A::A() Default constructor" << std::endl;
}

A::A(int i)
    : x(i)
{
    std::cout << "Hello from A::A(int) constructor" << std::endl;
}

A::A(const A& a)
    : x(a.x)
{
    std::cout << "Hello from A Copy constructor" << std::endl;
}

A::~A()
{
    std::cout << "Hello from A Destructor" << std::endl;
}

Program vector.cc
int main()
{
    std::cout << "Creating A Vector"; getchar();
    AVec_t av0;

    std::cout << "Adding an three elements to av0"; getchar();
    av0.push_back(A(2));   // Elements are appended using "push_back"
    std::cout << "After first push_back"; getchar();
    av0.push_back(A(10)); // Elements are appended using "push_back"
    std::cout << "After second push_back"; getchar();
    av0.push_back(A(100)); // Elements are appended using "push_back"
    // Number of elements in a vector can be queried with "size()"
    std::cout << "After third push_back, size av0 is " << av0.size() << std::endl;

    // Individual elements can be accessed with the [] operator
    std::cout << "Accessing elements with the [] operator"; getchar();
    std::cout << "av0[0].x is " << av0[0].x << std::endl;
    std::cout << "av0[1].x is " << av0[1].x << std::endl;
    std::cout << "av0[2].x is " << av0[2].x << std::endl;

    // Front and back of list have special accessors
    std::cout << "Accessing elements with the front and back"; getchar();
    std::cout << "av0.front().x is " << av0.front().x << std::endl;
    std::cout << "av0.back().x is " << av0.back().x << std::endl;

    // Vectors can be copied with copy constructor or assignment operator
    std::cout << "Making a copy of av0"; getchar();
    AVec_t av1(av0);
    std::cout << "Size of av1 is " << av1.size() << std::endl;
    std::cout << "av1[0].x is " << av1[0].x << std::endl;

    // Vectors can be shrunk with "pop_back". Notice that pop_back
    // does NOT return the element being popped
    std::cout << "Shrinking av0 with pop_back"; getchar();
    av0.pop_back();   // Remove last element
    std::cout << "Size of av0 is " << av0.size() << std::endl;
    av0.pop_back();   // Remove another element
    std::cout << "Size of av0 is " << av0.size() << std::endl;

    // Vectors can be initialized to "n" copies of a specified object
    std::cout << "Constructing AVec_t with 10 elements"; getchar();
    AVec_t av2(10, A(1)); // Makes 10 elements of A(1)
    std::cout << "Size of av2 is " << av2.size() << std::endl;
    std::cout << "av2[0].x is " << av2[0].x << std::endl;

    // All elements of a vector can be removed with "clear()"
    std::cout << "Clearing av2"; getchar();
    av2.clear();
    std::cout << "Size of av2 is " << av2.size() << std::endl;

    // Create and populate a vector of A pointers
    std::cout << "Creating A Pointer Vector"; getchar();
    APVec_t apv0;

    std::cout << "Adding an three elements to apv0"; getchar();

    Program vector.cc (continued)
apv0.push_back(new A(2));
apv0.push_back(new A(10));
apv0.push_back(new A(100));

// Number of elements in a vector can be queried with "size()"
std::cout << "Size of apv0 is " << apv0.size() << std::endl;

// Clear the apv0 vector. Note: ~A() NOT called. Why not?
apv0.clear();
std::cout << "Clearing apv0"; getchar();
apv0.clear();
std::cout << "Size of apb0 is " << apv0.size() << std::endl;

std::cout << "Main program exiting"; getchar();
return 0;
}

// Output from this program is:
// Creating A Vector
// Adding an three elements to av0
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Destructor
// After first push_back
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Hello from A Destructor
// After second push_back
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// After third push_back, size av0 is 3
// Accessing elements with the [] operator
// av0[0].x is 2
// av0[1].x is 10
// av0[2].x is 100
// Accessing elements with the front and back
// av0.front().x is 2
// av0.back().x is 100
// Making a copy of av0
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Size of av1 is 3
// av1[0].x is 2
// Shrinking av0 with pop_back
// Hello from A Destructor
// Size of av0 is 2
// Hello from A Destructor
// Size of av0 is 1

Program vector.cc (continued)
// Constructing AVec_t with 10 elements
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Size of av2 is 10
// av2[0].x is 1
// Clearing av2
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Size of av2 is 0
// Creating A Pointer Vector
// Adding an three elements to apv0
// Hello from A::A(int) constructor
// Hello from A::A(int) constructor
// Hello from A::A(int) constructor
// Size of apv0 is 3
// Clearing apv0
// Size of apb0 is 0
// Main program exiting
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Program vector.cc (continued)