```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    // begin() – Returns an iterator pointing to the first element in the vector
    // end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector
    // rbegin() – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
    // rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)
    // cbegin() – Returns a constant iterator pointing to the first element in the vector.
    // cend() – Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.
    // crbegin() – Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element
    // crend() – Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

    vector< int > g1;

    for (int i = 1; i <= 5; i++)
        g1.push_back(i);

    cout << "Output of begin and end: ";
    for (auto i = g1.begin(); i != g1.end(); ++i)
        cout << *i << " ";

    cout << "\nOutput of cbegin and cend: ";
    for (auto i = g1.cbegin(); i != g1.cend(); ++i)
        cout << *i << " ";

    cout << "\nOutput of rbegin and rend: ";
    for (auto ir = g1.rbegin(); ir != g1.rend(); ++ir)
        cout << *ir << " ";

    cout << "\nOutput of crbegin and crend : ";
    for (auto ir = g1.crbegin(); ir != g1.crend(); ++ir)
        cout << *ir << " ";

    //
    // size() – Returns the number of elements in the vector.
    // max_size() – Returns the maximum number of elements that the vector can hold.
    // capacity() – Returns the size of the storage space currently
```

```
allocated to the vector expressed as number of elements.
    // resize() – Resizes the container so that it contains ‘g’ elements.
    // empty() – Returns whether the container is empty.
    // shrink_to_fit() – Reduces the capacity of the container to fit its
size and destroys all elements beyond the capacity.
    // reserve() – Requests that the vector capacity be at least enough to
contain n elements.
    //

    vector< int > g2;

    for (int i = 1; i <= 5; i++)
        g2.push_back(i);

    cout << "Size : " << g2.size();
    cout << "\nCapacity : " << g2.capacity();
    cout << "\nMax_Size : " << g2.max_size();

    // resizes the vector size to 4
    g2.resize(4);

    // prints the vector size after resize()
    cout << "\nSize : " << g2.size();

    // checks if the vector is empty or not
    if (g2.empty() == false)
        cout << "\nVector is not empty";
    else
        cout << "\nVector is empty";

    // Shrinks the vector
    g2.shrink_to_fit();
    cout << "\nVector elements are: ";
    for (auto it = g2.begin(); it != g2.end(); it++)
        cout << *it << " ";

    // reference operator [g] – Returns a reference to the element at
position ‘g’ in the vector
    // at(g) – Returns a reference to the element at position ‘g’ in the
vector
    // front() – Returns a reference to the first element in the vector
    // back() – Returns a reference to the last element in the vector
    // data() – Returns a direct pointer to the memory array used
internally by the vector to store its owned elements.

    vector< int > g3;

    for (int i = 1; i <= 10; i++)
        g3.push_back(i * 10);

    cout << "\nReference operator [g] : g3[2] = " << g3[2];

    cout << "\nat : g3.at(4) = " << g3.at(4);
    cout << "\nfront() : g3.front() = " << g3.front();

    cout << "\nback() : g3.back() = " << g3.back();

    // pointer to the first element
    int* pos = g3.data();

    cout << "\nThe first element is " << *pos;

    // assign() – It assigns new value to the vector elements by replacing
old ones
    // push_back() – It push the elements into a vector from the back
    // pop_back() – It is used to pop or remove elements from a vector
from the back.
    // insert() – It inserts new elements before the element at the
specified position
    // erase() – It is used to remove elements from a container from the
specified position or range.
    // swap() – It is used to swap the contents of one vector with another
vector of same type and size.
    // clear() – It is used to remove all the elements of the vector
container
    // emplace() – It extends the container by inserting new element at
position
    // emplace_back() – It is used to insert a new element into the vector
container, the new element is added to the end of the vector

    // Assign vector
    vector< int > v;

    // fill the array with 10 five times
    v.assign(5, 10);

    cout << "The vector elements are: ";
    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";

    // inserts 15 to the last position
    v.push_back(15);
    int n = v.size();
    cout << "\nThe last element is: " << v[n - 1];

    // removes last element
    v.pop_back();

    // prints the vector
    cout << "\nThe vector elements are: ";
    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";

    // inserts 5 at the beginning
    v.insert(v.begin(), 5);

    cout << "\nThe first element is: " << v[0];
    // removes the first element
    v.erase(v.begin());

    cout << "\nThe first element is: " << v[0];

    // inserts at the beginning
    v.emplace(v.begin(), 5);
    cout << "\nThe first element is: " << v[0];

    // Inserts 20 at the end
    v.emplace_back(20);
    n = v.size();
    cout << "\nThe last element is: " << v[n - 1];

    // erases the vector
    v.clear();
    cout << "\nVector size after erase(): " << v.size();

    // two vector to perform swap
    vector< int > v1, v2;
    v1.push_back(1);
    v1.push_back(2);
    v2.push_back(3);
    v2.push_back(4);

    cout << "\n\nVector 1: ";
    for (int i = 0; i < v1.size(); i++)
        cout << v1[i] << " ";

    cout << "\nVector 2: ";
    for (int i = 0; i < v2.size(); i++)
        cout << v2[i] << " ";

    // Swaps v1 and v2
    v1.swap(v2);

    cout << "\nAfter Swap \nVector 1: ";
    for (int i = 0; i < v1.size(); i++)
        cout << v1[i] << " ";

    cout << "\nVector 2: ";
    for (int i = 0; i < v2.size(); i++)
        cout << v2[i] << " ";
    return 0;
}
```