```cpp
/*
    Salcedo, Salvador

    CS A250
    April 6, 2019

    Lab 9
*/

#include <iostream>
#include <string>
#include <vector>
#include <list>
#include <set>
#include <map>

using namespace std;

// Declaration function printVector.
// The function passes a vector and prints all
// the elements on one line, separated by a space.
// Use an iterator and a FOR loop.
void printVector(const vector<int>& a);

// Declaration function printList.
// The function passes a list and prints all
// the elements on one line, separated by a space.
// Use an iterator and a WHILE loop.
void printList(const list<int>& a);

// Declaration function printSet.
// The function passes a list and prints all
// the elements on one line, separated by a space.
// Use a range-based FOR loop. If you do not know
// what this means, search the Web.
void printSet(const set<int>& a);

int main()
{

    /
      ************************************************************************
      **
            VECTORS
    ***************************************************************************
      /
    cout << "  ***  STL VECTOR CLASS  ***  \n\n";

    // Use the default constructor to declare an integer vector v1.
    vector<int> v1;

    // void push_back (const value_type& val);
    // Use function push_back to insert the following values in v1: 12, 73, 41,
    // 38, 25, 56, an 63 in this order.
    v1.push_back(12);
    v1.push_back(73);
    v1.push_back(41);
    v1.push_back(38);
    v1.push_back(25);
    v1.push_back(56);
    v1.push_back(63);

    // size_type size() const noexcept;
    // Create a variable of type int named sizeV1 and store the size of the
      vector.
    // Use function size to retrieve the size of the vector.
    // Make sure you cast the return value of the function size to the
      appropriate type.
    int sizeV1 = (int)v1.size();

    // Use a FOR loop to print out the vector.
    // Do NOT use an iterator.
    for (int i = 0; i < sizeV1; i++) {
        cout << v1[i] << " ";
    }
    cout << endl;

    //void clear() noexcept;
    // Call the function clear on vector v1.
    v1.clear();

    // size_type size() const noexcept;
    // Call function size to print the size of v1.
    cout << v1.size() << endl;

    // size_type capacity() const noexcept;
    // Call function capacity to output the capacity of v1.
    cout << v1.capacity() << endl;

    // Create an array of integers containing: 10,11,12,13,14,15,16,17,18,19
    int a[] = {10,11,12,13,14,15,16,17,18,19};

    // Use the default constructor to declare an integer vector v2.
    vector<int> v2;

    // void assign (InputIterator first, InputIterator last);
    // Use function assign to copy elements 12, 13, 14, 15, and 16 in v2.
    // One statement only.
```

```cpp
    v2.assign(a + 2, a + 7);

    // Call the function printVector to print v2.
    printVector(v2);

    // const_reference back() const;
    // Use the function back to output the last element in the vector
    // (Notice that the back function returns a reference.)
    cout << v2.back() << endl;

    // Use the default constructor to declare an integer vector v3.
    vector<int> v3;

    // void assign (size_type n, const value_type& val);
    // Use function assign to insert the values 7, 7, 7, 7, and 7.
    // One statement only.
    v3.assign(5, 7);

    // Call the function printVector to print v3.
    printVector(v3);

    // const_reference at(size_type n) const;
    // Use function at to replace the middle element with 100.
    // (Notice that the at function returns a reference.)
    v3.at(2) = 100;

    // Call the function printVector to print v3.
    printVector(v3);

    // vector (const vector& x);
    // Use the copy constructor to create a new vector v4 with the
    // same elements of v3.
    vector<int> v4 = vector<int>(v3);

    // Call the function printVector to print v4.
    printVector(v4);

    // Create an iterator iterVector4 to point to the first element of v4.
    vector<int>::iterator iterVector4 = v4.begin();

    // Create an iterator iterVector2 to point to the second element of v2.
    vector<int>::iterator iterVector2 = v2.begin() + 1;

    // iterator insert (const_iterator position, InputIterator first,
      InputIterator last);
    // Use function insert to insert the second, third, and fourth element
    // of v2 as the first, second, and third element of v4.
    // (Notice that the insert function returns an iterator,
    //   but if we do not intend to use it, we can ignore it.)
    v4.insert(iterVector4, iterVector2, v2.begin() + 4);

    // Call the function printVector to print v4.
    printVector(v4);

    // iterator insert (const_iterator position, size_type n, const value_type& 
       val);
    // Use the function insert to insert three 0s at the end of v4.
    // (Notice that the insert function returns an iterator,
    //   but if we do not intend to use it, we can ignore it.)
    v4.insert((v4.begin() + v4.size()), 3, 0);

    // Call the function printVector to print v4.
    printVector(v4);

    // bool empty() const noexcept;
    // const_reference back() const;
    // void pop_back();
    // Use a WHILE loop to remove and output each element of v2 backwards.
    // Use function empty for the loop condition, function back to output
    // the last element, and function pop_back to remove elements.
    // (Notice that the insert function returns an iterator,
    //   but if we do not intend to use it, we can ignore it.)
    while (!v2.empty()) {
        cout << v2.back() << " ";
        v2.pop_back();
    }
    cout << endl;

    // void resize (size_type n, const value_type& val);
    // Use function resize to insert three times number 4 in v2.
    v2.resize(3, 4);

    // Call the function printVector to print v2.
    printVector(v2);

    // const_reference front() const;
    // Use function front to output the first element in v4.
    // (Notice that the front function returns a reference.)
    cout << v4.front() << endl;

    // void swap (vector& x);
    // Use function swap to swap v2 with v4.
    v2.swap(v4);

    // Call the function printVector to print v2.
    printVector(v2);

    // Create a new vector v5;
    vector<int> v5;

    // Use the overloaded assignment operator to copy all the elements of v2
    // into v5.
    v5 = v2;

    // void resize (size_type n);
    // size_type size() const noexcept;
    // Delete the last element of v5 by using the functions resize and size
    v5.resize((int)v5.size() - 1);

    // Call the function printVector to print v5.
    printVector(v5);

    // Create an iterator iterVector5 to point to the first element of v5.
    vector<int>::iterator iterVector5 = v5.begin();

    // iterator erase (const_iterator first, const_iterator last);
    // size_type size() const noexcept;
    // Call the function erase to delete the second half of v5.
    // Use function size to get the range.
    // (Notice that the insert function returns an iterator,
    //   but if we do not intend to use it, we can ignore it.)
    v5.erase(v5.begin() + (v5.size() / 2), v5.begin() + v5.size());

    // Call the function printVector to print v5 again.
    printVector(v5);

    // iterator erase (const_iterator position);
    // Call the function erase to delete the first element of the vector.
    // (Notice that the insert function returns an iterator,
    //   but if we do not intend to use it, we can ignore it.)
    v5.erase(v5.begin());

    // Call the function printVector to print v5 again.
    printVector(v5);

    // Create a vector of integers named v6 containing numbers from 100 to 105.
    // Using the copy constructor, create a vector named v7, a copy of v6.
    vector<int> v6 = { 100, 101, 102, 103, 104, 105 };
    vector<int> v7 = vector<int>(v6);

    // iterator erase (const_iterator position);
    // iterator insert (const_iterator position, const value_type& val);
    // Erase element 103 from v7 and insert element 333 in its plase,
    // by using an iterator.
    // Note that the function erase returns an iterator that can be used
    // to insert 333 in the right position.
    v7.insert(v7.erase(v7.begin() + 3), 333);
```

```cpp

    // Using a range-based FOR loop, print v7.
    for (int i : v7) {
        cout << i << " ";
    }
    cout << endl;

    /
      *************************************************************************
      **
              LISTS
    *************************************************************************
      /

    cout << "\n-----------------------------------------------------";
    cout << "\n  ***  STL LIST CLASS  ***  \n\n\n";

    // Use the default constructor to create three lists of integers, intLis1,
    // intList2, and intList3.
    list<int> intList1;
    list<int> intList2;
    list<int> intList3;

    // void push_back (const value_type& val);
    // Use the function push_back to insert the following values in the first
      list:
    // 23 58 58 58 36 15 15 93 98 58
    int b[] = { 23, 58, 58, 58, 36, 15, 15, 93, 98, 58 };
    for (int i = 0; i < 10; i++) {
        intList1.push_back(b[i]);
    }

    // Call function printList to print intList1.
    printList(intList1);

    // Using the overloaded assignment operator, copy elements of intList1 into
       intList2.
    intList2 = intList1;

    // Call function printList to print intList2.
    printList(intList2);

    // void sort();
    // Using function sort, sort all elements in the second list.
    intList2.sort();

    // Call function printList to print intList2.
    printList(intList2);

    // void unique();
    // Using function unique, removes all consecutive duplicates in the list.
    intList2.unique();

    // Call function printList to print intList2.
    printList(intList2);

    // void push_back (const value_type& val);
    //Insert the following elements in the third list:
    //  13 23 25 136 198
    int c[] = { 13, 23, 25, 136, 198 };
    for (int i = 0; i < 5; i++) {
        intList3.push_back(c[i]);
    }

    // Call function printList to print intList3.
    printList(intList3);

    // void merge (list& x);
    // Add to the second list all elements of the third list(browse the
    //  list of functions in cplusplus.com to figure out which function
    //  you need to use).
    // --> This is ONE statement only.
    intList3.merge(intList2);

    // Call function printList to print intList2.
    printList(intList3);

    /
      ************************************************************************
      **
            SETS
    ************************************************************************
      /
    cout << "\n\n----------------------------------------------------";
    cout << "\n  ***  STL SET CLASS  ***  \n\n";

    // Create a set of ints named set1 using the initializer list to insert the

    // following integers in this order: 2, 7, 5, 6, 9, 1 and 3.
    set<int> set1{ 2, 7, 5, 6, 9, 1, 3 };

    // Print the set using the printSet function you implemented.
    printSet(set1);

    // What do you notice in the printout?
        //The items were arranged in ascending order
    // size_type  erase (const value_type& val);
    // Use the function erase integer 9 from set1.
    // Print out set1.
    set1.erase(9);
    printSet(set1);

    // size_type  erase (const value_type& val);
    // Use the function erase integer 2 from set1, but
    // this time use cout to print the return value.
    // What is the return value?
    cout << set1.erase(2); //it returns a reference to the value before it, in 
      this case it was 1

    // If you do not know what the return value is, then
    // check set::erase in cplusplus.com
    cout << endl;

    // Print set1.
    printSet(set1);

    // iterator  erase (const_iterator position);
    // This function is different from the previous one,
    // because instead of passing a value, it passes a
    // position indicated by an iterator.
    // Delete the second element in the set without creating
    // an iterator variable and using the prefix increment
    // operator.
    set1.erase(++set1.begin());

    // Print set1.
    printSet(set1);

    // pair<iterator,bool> insert (const value_type& val);
    // Use the function insert to insert 4 and 8 in set1.
    set1.insert(4);
    set1.insert(8);

    // Print set1.
    printSet(set1);

    set<int>::iterator first = set1.begin();
    set<int>::iterator second = ++set1.begin();
    // iterator  erase(const_iterator first, const_iterator last);
    // Use function erase to delete the first and second element
    // in set1. Use the given iterators created above.
    // Note that you should write one statement only.
    set1.erase(first, ++second);

    // Print set1.
    printSet(set1);

```

```cpp
    // Your output should be: 5 6 7 8
    // If it is not, you need to carefully view the function erase
    // to understand how it works.

    cout << "\n\n---------------------------------------------------";
    cout << "\n\nThe output for the next sections depends on your implementation.";

    /***************************************************************************
    MAPS
    ***************************************************************************/
    cout << "\n\n---------------------------------------------------";
    cout << "\n  ***  STL MAP CLASS  ***  \n\n";

    // Create a few maps using the different constructors shown in the slides.
    // Use the following functions to manipulate the maps:
    // pair<iterator,bool> insert (const value_type& val);
    // void insert (InputIterator first, InputIterator last);
    // iterator  erase(const_iterator position);
    // size_type erase(const key_type& k);
    // iterator  erase(const_iterator first, const_iterator last);
    // Print each map without creating a print function, but by using
    // a loop.

    /*Map 1*/
    map<int, string> map1;
    for (int i = 0; i < 10; i++) {
        string d[] = { "null", "first", "second", "third", "fourth", "fifth",
            "sixth", "seventh", "eight", "nineth"};

        map1.insert(make_pair(i, d[i]));
    }

    map<int, string>::const_iterator mIter1 = map1.cbegin();
    for (mIter1; mIter1 != map1.cend(); mIter1++) {
        cout << "(" << mIter1->first << ", " << mIter1->second << ")" << endl;
    }

    cout << "\n";

    /*Map 2*/
    map<int, string> map2;
    for (int i = 1; i < 10; i++) {
        string d[] = { "A", "B", "C", "D", "E", "F",
            "G", "H", "I", "J" };

        map2.insert(make_pair(i, d[i - 1]));
    }

    map<int, string>::const_iterator mIter2 = map2.cbegin();
    for (mIter2; mIter2 != map2.cend(); mIter2++) {
        cout << "(" << mIter2->first << ", " << mIter2->second << ")" << endl;
    }

    /**************************************************************************
    *       Create statements using the functions below.
    *       You might need to create new containers.
    *       Is the output what you expected?
    **************************************************************************/


    cout << "\n\n-----------------------------------------------------";
    cout << "\n  ***  OTHER FUNCTIONS  ***  \n\n";

    // list: void assign (size_type n, const value_type& val);

    // vector: void assign (InputIterator first, InputIterator last);

    // list: const_reference back() const;
    // (Notice that this back function returns a reference.)

    // list: void clear() noexcept;

    // list: bool empty() const noexcept;

    // vector: const_reference front() const;

    // list: iterator insert (const_iterator position, const value_type& val);
    // (Notice that the insert function returns an iterator.)

    // list: void pop_back();

    // list: void pop_front();

    // list: void push_front (const value_type& val);

    // list: void remove (const value_type& val);

    // list: void reverse() noexcept;

    // list: void splice (const_iterator position, list& x);

    // list: void splice (const_iterator position, list& x, const_iterator i);

    // list: void splice (const_iterator position, list& x, const_iterator
        first, const_iterator last);

    // set: void swap (set& x);

    // set: const_iterator find (const value_type& val) const;

    cout << "\n\n----------------------------------------------------";

    cout  <<  endl;
    system("Pause");
    return 0;
}

// Definition function printVector
void printVector(const vector<int>& a) {
    vector<int>::const_iterator iter = a.cbegin();

    for (iter; iter != a.cend(); iter++) {
        cout << *iter << " ";
    }
    cout << endl;
}

// Definition function printList
void printList(const list<int>& a) {
    list<int>::const_iterator iter = a.cbegin();

    while (iter != a.cend()) {
        cout << *iter << " ";
        iter++;
    }
    cout << endl;
}

// Definition function printSet
void printSet(const set<int>& a) {
    set<int>::const_iterator iter = a.cbegin();

    for (iter; iter != a.cend(); iter++) {
        cout << *iter << " ";
    }
    cout << endl;
}
```