```
#include <iostream>
#include <vector>
#include <list>
#include <algorithm> // for transform()
#include <functional>
#include <iterator> // for back_inserter() and ostream_iterator()

#include <cstdlib> // EXIT_SUCCESS

using namespace std;


int Add5(int a) { return a + 5; }

int AddInts(int arg1, int arg2) { return arg1 + arg2; }


// Illustration of the usage of binders (bind1st and bind2nd)
// and adapters e.g. ptr_fun or mem_fun_ref.
// Note that most of these functions seems to be deprecated in the new
// coming standard of C++ (C++ - 11).
int main(void) {
  // Fill an input vector to be used in the example
  vector<int> v;
  v.push_back(1); v.push_back(5); v.push_back(4); v.push_back(2);


  // For storing the result of the mapping
  // Remark: we could also have written directly to std output (see
  // example 2 with the list of words)
```

```
    vector<int> result;

    cout << "----------------------" << endl;
    cout << "Add 5 to integers in a container:" << endl;


    // Version 1 with function pointer
    cout << "Version 1: with a function pointer"
         << endl;

    transform(v.begin(), v.end(), back_inserter(result), Add5);

    // Print the result to std output
    copy(result.begin(), result.end(),
         ostream_iterator<int>(cout, "\n"));


    // Version 2 with generic function of two arguments
    // and the second argument binded to 5
    cout << "Version 2: with generic function"
         << " of two arguments"
         << endl;

    result.clear();
    transform(v.begin(), v.end(), back_inserter(result),
              bind1st(ptr_fun(AddInts), 5));

    copy(result.begin(), result.end(),
         ostream_iterator<int>(cout, "\n"));
```

```
  cout << "--------------------" << endl;
  cout << "Compute the length of strings in a string container"
       << " by mapping a member function"
       << endl;

  // Compute the length of each element of a container and
  // write the result on standard output

  list<string> words;
  words.push_back("a");
  words.push_back("list");
  words.push_back("of");
  words.push_back("words");

  transform(words.begin(), words.end(),
            // write the result directly to std output
            ostream_iterator<int>(cout, "\n"),
            // adapt a member function to a function object
            mem_fun_ref(&string::length)
            );

  return(EXIT_SUCCESS);
}
```