```
// FILE: Assign03.cpp
// An interactive test program for sequence class
#include <cctype>      // provides toupper
#include <iostream>    // provides cout and cin
#include <cstdlib>     // provides EXIT_SUCCESS
#include "Sequence.h"  // with value_type defined as double
using namespace std;
using namespace CS3358_FA2019;

// PROTOTYPES for functions used by this test program:
void print_menu();
// Pre: (none)
// Post: A menu of choices for this program is written to cout.

char get_user_command();
// Pre: (none)
// Post: The user has been prompted to enter a one character command.
//   The next character has been read (skipping blanks and newline
//   characters), and this character is returned.  The input buffer
//   is cleared of any extra input until and including the first
//   newline character.

void show_sequence(sequence src);
// Pre: (none)
// Post: The items of src are displayed on cout (one per line).
/* Quiz: Why isn't src passed by const reference? */

double get_number();
// Pre: (none)
// Post: The user has been prompted to enter a real number. The prompt
//   is repeated until a valid real number can be read. The valid real
//   number has been read is returned. The input buffer is cleared of
//   any extra input until and including the first newline character.

int main()
{
   sequence test; // sequence to perform tests on
   char choice;   // command character entered by user

   cout << "I have initialized an empty sequence of real numbers."
        << endl;

   do
   {
      print_menu();
      choice = toupper( get_user_command() );
      switch (choice)
      {
      case '!':
         test.start();
         break;
      case '+':
         test.advance();
         break;
      case '?':
         if (test.is_item())
            cout << "There is a current item." << endl;
         else
            cout << "There is no current item." << endl;
         break;
      case 'C':
         if (test.is_item())
            cout << "Current item is: " << test.current() << endl;
         else
            cout << "There is no current item." << endl;
         break;
      case 'P':
         show_sequence(test);
         break;
      case 'S':
         cout << "Size is " << test.size() << '.' << endl;
         break;
      case 'I':
         test.insert(get_number());
         break;
      case 'A':
         test.attach(get_number());
         break;
      case 'R':
         test.remove_current();
            cout << "The current item has been removed." << endl;
         break;
      case 'Q':
         cout << "Quit option selected...terminating..." << endl;
         break;
      default:  cout << choice << " is invalid." << endl;
      }
   }
   while (choice != 'Q');

   cout << "Press Enter or Return when ready...";
   cin.get();
   return EXIT_SUCCESS;
}

void print_menu()
{
   cout << endl;
   cout << "Following choices are available: " << endl;
   cout << "  !  Activate start() function" << endl;
   cout << "  +  Activate advance() function" << endl;
   cout << "  ?  Print result from is_item() function" << endl;
   cout << "  C  Print result from current() function" << endl;
   cout << "  P  Print a copy of entire sequence" << endl;
   cout << "  S  Print result from size() function" << endl;
   cout << "  I  Insert a new number with insert(...) function" << 
endl;
   cout << "  A  Attach a new number with attach(...) function" << 
endl;
   cout << "  R  Activate remove_current() function" << endl;
   cout << "  Q  Quit this test program" << endl;
}

char get_user_command()
{
   char command;

   cout << "Enter choice: ";
   cin >> command;
   cin.ignore(999, '\n');

   return command;
}

void show_sequence(sequence src)
{
   for ( src.start(); src.is_item(); src.advance() )
      cout << src.current() << endl;
}

double get_number()
{
   double result;

   cout << "Enter a real number: ";
   cin  >> result;
   while ( ! cin.good() )
   {
     cerr << "Invalid real number input..."<< endl;
     cin.clear();
     cin.ignore(999, '\n');
     cout << "Re-enter real number ";
     cin  >> result;
   }
   cin.ignore(999, '\n');

   return result;

}
```