```cpp
// FILE: sequenceTest.cpp
// An interactive test program for the sequence class

#include <cctype>      // provides toupper
#include <iostream>    // provides std::cout and cin
#include <cstdlib>     // provides EXIT_SUCCESS
#include "sequence.h"
namespace seqOfNum  = CS3358_SP2023_A04_sequenceOfNum;
namespace seqOfChar = CS3358_SP2023_A04_sequenceOfChar;
using namespace std;

// PROTOTYPES for functions used by this test program:

void print_menu();
// Pre:  (none)
// Post: A menu of choices for this program is written to std::cout.
char get_user_command();
// Pre:  (none)
// Post: The user is prompted to enter a one character command.
//       The next character is read (skipping blanks and newline
//       characters), and this character is returned.
void show_list(seqOfNum::sequence<double> src);
// Pre: (none)
// Post: The items of src are printed to std::cout (one per line).
void show_list(seqOfChar::sequence<char> src);
// Pre: (none)
// Post: The items of src are printed to std::cout (one per line).
int get_object_num();
// Pre:  (none)
// Post: The user is prompted to enter either 1 or 2. The
//       prompt is repeated until a valid integer can be read
//       and the integer's value is either 1 or 2. The valid
//       integer read is returned. The input buffer is cleared
//       of any extra input until and including the first
//       newline character.
double get_number();
// Pre:  (none)
// Post: The user is prompted to enter a real number. The prompt
//       is repeated until a valid real number can be read. The
//       valid real number read is returned. The input buffer is
//       cleared of any extra input until and including the
//       first newline character.
char get_character();
// Pre:  (none)
// Post: The user is prompted to enter a non-whitespace character.
//       The prompt is repeated until a non-whitespace character
//       can be read. The non-whitespace character read is returned.
//       The input buffer is cleared of any extra input until and
//       including the first newline character.
```

```
50
51 int main(int argc, char *argv[])
52 {
53    seqOfNum::sequence<double> s1;  // A sequence of double for testing
54    seqOfChar::sequence<char> s2; // A sequence of char for testing
55    int objectNum;    // A number to indicate selection of s1 or s2
56    double numHold;   // Holder for a real number
57    char charHold;    // Holder for a character
58    char choice;      // A command character entered by the user
59
60    std::cout << "An empty sequence of real numbers (s1) and\n"
61         << "an empty sequence of characters (s2) have been created."
62         << endl;
63
64    do
65    {
66       if (argc == 1)
67          print_menu();
68       choice = toupper( get_user_command() );
69       switch (choice)
70       {
71          case '!':
72             objectNum = get_object_num();
73             if (objectNum == 1)
74             {
75                s1.start();
76                std::cout << "s1 started" << endl;
77             }
78             else
79             {
80                s2.start();
81                std::cout << "s2 started" << endl;
82             }
83             break;
84          case '&':
85             objectNum = get_object_num();
86             if (objectNum == 1)
87             {
88                s1.end();
89                std::cout << "s1 ended" << endl;
90             }
91             else
92             {
93                s2.end();
94                std::cout << "s2 ended" << endl;
95             }
96             break;
97          case '+':
98             objectNum = get_object_num();
```

```
 99             if (objectNum == 1)
100             {
101                if ( ! s1.is_item() )
102                   std::cout << "Can't advance s1." << endl;
103                else
104                {
105                   s1.advance();
106                   std::cout << "Advanced s1 one item."<< endl;
107                }
108             }
109             else
110             {
111                if ( ! s2.is_item() )
112                   std::cout << "Can't advance s2." << endl;
113                else
114                {
115                   s2.advance();
116                   std::cout << "Advanced s2 one item."<< endl;
117                }
118             }
119             break;
120          case '-':
121             objectNum = get_object_num();
122             if (objectNum == 1)
123             {
124                if ( ! s1.is_item() )
125                   std::cout << "Can't move back s1." << endl;
126                else
127                {
128                   s1.move_back();
129                   std::cout << "Moved s1 back one item."<< endl;
130                }
131             }
132             else
133             {
134                if ( ! s2.is_item() )
135                   std::cout << "Can't move back s2." << endl;
136                else
137                {
138                   s2.move_back();
139                   std::cout << "Moved s2 back one item."<< endl;
140                }
141             }
142             break;
143          case '?':
144             objectNum = get_object_num();
145             if (objectNum == 1)
146             {
147                if ( s1.is_item() )
```

```
148                     std::cout << "s1 has a current item." << endl;
149                  else
150                     std::cout << "s1 has no current item." << endl;
151               }
152               else
153               {
154                  if ( s2.is_item() )
155                     std::cout << "s2 has a current item." << endl;
156                  else
157                     std::cout << "s2 has no current item." << endl;
158               }
159               break;
160            case 'C':
161               objectNum = get_object_num();
162               if (objectNum == 1)
163               {
164                  if ( s1.is_item() )
165                     std::cout << "Current item in s1 is: "
166                          << s1.current() << endl;
167                  else
168                     std::cout << "s1 has no current item." << endl;
169               }
170               else
171               {
172                  if ( s2.is_item() )
173                     std::cout << "Current item in s2 is: "
174                          << s2.current() << endl;
175                  else
176                     std::cout << "s2 has no current item." << endl;
177               }
178               break;
179            case 'P':
180               objectNum = get_object_num();
181               if (objectNum == 1)
182               {
183                  if (s1.size() > 0)
184                  {
185                     std::cout << "s1: ";
186                     show_list(s1);
187                     std::cout << endl;
188                  }
189                  else
190                     std::cout << "s1 is empty." << endl;
191               }
192               else
193               {
194                  if (s2.size() > 0)
195                  {
196                     std::cout << "s2: ";
```

```
197                     show_list(s2);
198                     std::cout << endl;
199                  }
200                  else
201                     std::cout << "s2 is empty." << endl;
202               }
203               break;
204            case 'S':
205               objectNum = get_object_num();
206               if (objectNum == 1)
207                  std::cout << "Size of s1 is: " << s1.size() << endl;
208               else
209                  std::cout << "Size of s2 is: " << s2.size() << endl;
210               break;
211            case 'A':
212               objectNum = get_object_num();
213               if (objectNum == 1)
214               {
215                  numHold = get_number();
216                  s1.add(numHold);
217                  std::cout << numHold << " added to s1." << endl;
218               }
219               else
220               {
221                  charHold = get_character();
222                  s2.add(charHold);
223                  std::cout << charHold << " added to s2." << endl;
224               }
225               break;
226            case 'R':
227               objectNum = get_object_num();
228               if (objectNum == 1)
229               {
230                  if ( s1.is_item() )
231                  {
232                     numHold = s1.current();
233                     s1.remove_current();
234                     std::cout << numHold << " removed from s1." << endl;
235                  }
236                  else
237                     std::cout << "s1 has no current item." << endl;
238               }
239               else
240               {
241                  if ( s2.is_item() )
242                  {
243                     charHold = s2.current();
244                     s2.remove_current();
245                     std::cout << charHold << " removed from s2." << endl;
```

```
246                }
247                else
248                   std::cout << "s2 has no current item." << endl;
249             }
250             break;
251          case 'Q':
252             std::cout << "Quit option selected...bye" << endl;
253             break;
254          default:
255             std::cout << choice << " is invalid...try again" << endl;
256       }
257    }
258    while (choice != 'Q');
259
260    cin.ignore(999, '\n');
261    std::cout << "Press Enter or Return when ready...";
262    cin.get();
263    return EXIT_SUCCESS;
264 }
265
266 void print_menu()
267 {
268    std::cout << endl;
269    std::cout << "The following choices are available:\n";
270    std::cout << "  !  Activate the start() function\n";
271    std::cout << "  &  Activate the end() function\n";
272    std::cout << "  +  Activate the advance() function\n";
273    std::cout << "  -  Activate the move_back() function\n";
274    std::cout << "  ?  Print the result from the is_item() function\n";
275    std::cout << "  C  Print the result from the current() function\n";
276    std::cout << "  P  Print a copy of the entire sequence\n";
277    std::cout << "  S  Print the result from the size() function\n";
278    std::cout << "  A  Add a new item with the add(...) function\n";
279    std::cout << "  R  Activate the remove_current() function\n";
280    std::cout << "  Q  Quit this test program" << endl;
281 }
282
283 char get_user_command()
284 {
285    char command;
286
287    std::cout << "Enter choice: ";
288    cin >> command;
289
290    std::cout << "You entered ";
291    std::cout << command << endl;
292    return command;
293 }
294
```

```
295 void show_list(seqOfNum::sequence<double> src)
296 {
297    for ( src.start(); src.is_item(); src.advance() )
298       std::cout << src.current() << "  ";
299 }
300
301 void show_list(seqOfChar::sequence<char> src)
302 {
303    for ( src.start(); src.is_item(); src.advance() )
304       std::cout << src.current() << "  ";
305 }
306
307 int get_object_num()
308 {
309    int result;
310
311    std::cout << "Enter object # (1 = s1, 2 = s2) ";
312    cin  >> result;
313    while ( ! cin.good() )
314    {
315       cerr << "Invalid integer input..." << endl;
316       cin.clear();
317       cin.ignore(999, '\n');
318       std::cout << "Re-enter object # (1 = s1, 2 = s2) ";
319       cin  >> result;
320    }
321    // cin.ignore(999, '\n');
322
323    while (result != 1 && result != 2)
324    {
325       cin.ignore(999, '\n');
326       cerr << "Invalid object # (must be 1 or 2)..." << endl;
327       std::cout << "Re-enter object # (1 = s1, 2 = s2) ";
328       cin  >> result;
329       while ( ! cin.good() )
330       {
331          cerr << "Invalid integer input..." << endl;
332          cin.clear();
333          cin.ignore(999, '\n');
334          std::cout << "Re-enter object # (1 = s1, 2 = s2) ";
335          cin  >> result;
336       }
337       // cin.ignore(999, '\n');
338    }
339
340    std::cout << "You entered ";
341    std::cout << result << endl;
342    return result;
343 }
```

```
344
345 double get_number()
346 {
347    double result;
348
349    std::cout << "Enter a real number: ";
350    cin  >> result;
351    while ( ! cin.good() )
352    {
353       cerr << "Invalid real number input..." << endl;
354       cin.clear();
355       cin.ignore(999, '\n');
356       std::cout << "Re-enter a real number ";
357       cin  >> result;
358    }
359    // cin.ignore(999, '\n');
360
361    std::cout << "You entered ";
362    std::cout << result << endl;
363    return result;
364 }
365
366 char get_character()
367 {
368    char result;
369
370    std::cout << "Enter a non-whitespace character: ";
371    cin  >> result;
372    while ( ! cin )
373    {
374       cerr << "Invalid non-whitespace character input..." << endl;
375       cin.ignore(999, '\n');
376       std::cout << "Re-enter a non-whitespace character: ";
377       cin  >> result;
378    }
379    // cin.ignore(999, '\n');
380
381    std::cout << "You entered ";
382    std::cout << result << endl;
383    return result;
384 }
385
```