```cpp
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C/C++ Programming IV : Advanced Programming with Objects
// 152488 Raymond L. Mitchell III
// hw1.cpp
// Win10
// Visual C++ 19.0
//

#include <iostream>
#include <algorithm>
#include <deque>
#include <string>
#include <vector>
#include <exception>
#include <iterator>        // contains back_inserter
#include <numeric>         // contains accumulate
#include <sstream>          // contains ostringstream

using namespace std;

// class generators:
struct UniqueNumbers {
   int current;
   UniqueNumbers() { current = 0; }
   int operator()() { return ++current; }
} uniqueNumber;

struct LowerCharUnique {
   char current;
   LowerCharUnique() { current = 'a'; }
   char operator()() { return current++; }
} uniqueLowerLetter;

// Parser Function:
bool isEven(int i) { return (i % 2) == 0; }

// Unit Tests:
void testDeque()
{
   const size_t NUM_VALUES = 10;
   const int  VALID_VALUE = 55;
   deque<int> tempDeque;

   // 1a) populate the testDeque with sequential values 1 to 10
   generate_n(back_inserter(tempDeque), NUM_VALUES, uniqueNumber);

   // 1b) add all values within the testDeque (initial value is 0)
   int testValue = accumulate(tempDeque.begin(), tempDeque.end(), 0);

   // 1c) Check if expected value is calculated
   if (testValue == VALID_VALUE)
```

```
53          clog << "testDeque PASSED\n";
54       else
55          clog << "testDeque FAILED : Expected accumulation "
56          << VALID_VALUE << " instead saw " << testValue << "\n";
57    }
58
59    void testString()
60    {
61       const size_t NUM_VALUES = 26;
62       string  tempString;
63       const string VALID_VALUE = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
64
65       // 2a) populate tempString with lowercase letters ascending order
66       generate_n(back_inserter(tempString), NUM_VALUES, uniqueLowerLetter);
67
68       // 2b) use transform to convert string to uppercase
69       transform(tempString.begin(), tempString.end(), tempString.begin(), toupper);
70
71       // 2c) Check if expected value is calculated (if true output = 0)
72       if (tempString.compare(VALID_VALUE) == 0)
73          clog << "testString PASSED\n";
74       else
75          clog << "testString FAILED : Expected string "
76          << VALID_VALUE << " instead saw " << tempString << "\n";
77    }
78
79    void testVector()
80    {
81       vector<int>::iterator divider;
82       const string VALID_VALUE = "24681013579";
83       ostringstream tempOStream;
84       std::ostream_iterator<int> out_it(tempOStream);
85
86       //3a) create an array literal containing the values 10 to 1
87       const int INIT_ARRAY[] = {10,9,8,7,6,5,4,3,2,1};
88
89       //3b) initialize vector to initArray using iterator range constructor
90       vector<int> tempVector(INIT_ARRAY, INIT_ARRAY + sizeof(INIT_ARRAY) / sizeof
          (int));
91
92       //3c) rearrange values to place even numbers in lower half of vector
93       divider = partition(tempVector.begin(), tempVector.end(), isEven);
94
95       //3d) sort on each partition
96       // sort even partition
97       sort(tempVector.begin(), divider);
98       // sort odd partition
99       sort(divider, tempVector.end());
100
101      // 3e) Copy vector to an ostring_stream
102      copy(tempVector.begin(), tempVector.end(), out_it);
103
```

```
104     // 3f) Verify ostring_stream contents
105     if (tempOStream.str().compare(VALID_VALUE) == 0)
106        clog << "testVector PASSED\n";
107     else
108        clog << "testVector FAILED : Expected output "
109        << VALID_VALUE << " instead saw " << tempOStream.str() << "\n";
110  }
111
112  int main(void)
113  {
114     //unit test demonstrating deque & algorithm functionality
115     testDeque();
116
117     //unit test demonstrating string & algorithm functionality
118     testString();
119
120     // unit test demonstrating vector & algorithm functionality
121     testVector();
122  }
123
```