```cpp
//
// Shaun Chemplavil U08713628
// shaun.chemplavil@gmail.com
// C/C++ Programming III : Intermediate Programming with Objects
// 151116  Raymond L. Mitchell III
// StringUtility.cpp
// Win10
// Visual C++ 19.0
//
// File contains the display member function for the StringUtility class
//

#include "StringUtility.h"

namespace ShaunChemplavil
{
   string StringUtility::join(const vector<string> &strings, char delimiter)
   {
      string out;
      for (size_t i = 0; i < strings.size(); i++)
      {
         //append string to output string
         out += strings[i];
         // Do not add the delimiter after the last string!
         if (i < (strings.size() - 1))
            out += delimiter;
      }
      return out;
   }
   vector<string> StringUtility::reverse(const vector<string> &strings)
   {
      vector<string> out;
      // Use an int to index each element of vector of strings
      // start with last element, and decrement until the index is
      // negative
      for (int i = (strings.size() - 1); i >= 0; i--)
      {
         // declare temporary string to append elements of string
         // indexing logic is the same as for the vector of strings
         string temp;
         for (int j = (strings[i].size() - 1); j >= 0; j--)
            temp += strings[i].at(j);

         // Add the reversed string onto the output vector of strings
         out.push_back(temp);
      }
      return out;
   }
   vector<string> StringUtility::combine(const vector<string> &left, const
    vector<string> &right)
   {
      vector<string> out;
```

```
52
53          //Append every permutation of the left and right input vectors of string
54          for (size_t i = 0; i < left.size(); i++)
55             for (size_t j = 0; j < right.size(); j++)
56                // append the elements of each vector of strings
57                out.push_back(left[i] + right[j]);
58
59          return out;
60       }
61       vector<string> StringUtility::leftPad(const vector<string> &strings, char pad)
62       {
63          vector<string> out;
64          size_t max_size = 0;
65          // Find the longest string in the vector
66          for (size_t i = 0; i < strings.size(); i++)
67             max_size = max_size > strings[i].size() ? max_size : strings[i].size();
68
69          // Now do appropriate left padding
70          for (size_t i = 0; i < strings.size(); i++)
71          {
72             string padStr;
73
74             out.push_back(padStr.assign(max_size - strings[i].size(), pad).append
                (strings[i]));
75          }
76
77          return out;
78       }
79    }
80
```