```
/*
 * StringUtility.h
 *
 *  Created on: Nov 18, 2013
 *      Author: Nathaniel Gallinger
 */

#ifndef STRINGUTILITY_H_
#define STRINGUTILITY_H_

#include <string>
using std::string;

#include <vector>
using std::vector;

namespace NathanielGallinger
{
  class StringUtility
  {
  public:
    string join(const vector<string> &str, char delim);
    vector<string> reverse(const vector<string> &str);
    vector<string> combine(const vector<string> &left, const vector<string> &right);
    vector<string> leftPad(const vector<string> &str, char pad);
  };
}

#endif /* STRINGUTILITY_H_ */


/*
 * StringUtility.cpp
 *
 *  Created on: Nov 18, 2013
 *      Author: Nathaniel Gallinger
 */

#include "StringUtility.h"

// Join Function
string
NathanielGallinger::StringUtility::join(const vector<string> &strVect, char delim)
{
  string retval;

  for (unsigned int idx = 0; idx < strVect.size(); idx++) {
    retval += strVect[idx];
    if ((idx + 1) < strVect.size())
      retval += delim;
  }

  return retval;
}

// Reverse Function
vector<string>
NathanielGallinger::StringUtility::reverse(const vector<string> &strVect)
{
  vector<string> retval;
```

```
  for (unsigned int idx = 0; idx < strVect.size(); idx++)
    retval.push_back(strVect[(strVect.size() - 1)- idx]);

  return retval;
}

// Combine Function
vector<string>
NathanielGallinger::StringUtility::combine(const vector<string> &left, const
vector<string> &right)
{
  vector<string> retval;

  for (unsigned int idx = 0; idx < left.size(); idx++)
    for (unsigned int idx2 = 0; idx2 < right.size(); idx2++)
      retval.push_back(left[idx] + right[idx2]);

  return retval;
}

// Leftpad Function
vector<string>
NathanielGallinger::StringUtility::leftPad(const vector<string> &strVect, char pad)
{
  vector<string> retval;
  unsigned int maxSize = 0;

  // Find max size
  for (unsigned int idx = 0; idx < strVect.size(); idx++) {
    if (strVect[idx].size() > maxSize)
      maxSize = strVect[idx].size();
  }

  // Add padding
  for (unsigned int idx = 0; idx < strVect.size(); idx++) {
    retval.push_back("");
    retval[idx].insert(0, maxSize - strVect[idx].size(), pad);
    retval[idx] += strVect[idx];
  }

  return retval;
}


/*
 * hw7.cpp
 *
 *  Created on: Nov 18, 2013
 *      Author: Nathaniel Gallinger
 */

#include "StringUtility.h"
#include <iostream>
using std::cout;

int main()
{
  // Sample input string
  vector<string> input;
```

```
  input.push_back("The");
  input.push_back("quick");
  input.push_back("brown");
  input.push_back("fox");
  input.push_back("jumps");
  input.push_back("over");
  input.push_back("the");
  input.push_back("lazy");
  input.push_back("dog");

  // Test join
  cout << "== Testing Join ==\n";
  char delim = ',';
  NathanielGallinger::StringUtility string_util;
  string test1 = string_util.join((const vector<string>)input, delim);
  cout << test1 << "\n";

  // Test reverse
  cout << "== Testing reverse ==\n";
  vector<string> test2 = string_util.reverse((const vector<string>)input);
  for (unsigned int idx = 0; idx < test2.size(); idx++)
    cout << test2[idx] << "\n";

  // Test combine
  cout << "== Testing combine ==\n";
  vector<string> left;
  left.push_back("Mr.");
  left.push_back("Mrs.");
  vector<string> right;
  right.push_back("Jones");
  right.push_back("Smith");
  right.push_back("Williams");
  vector<string> test3 = string_util.combine((const vector<string>)left,
                                             (const vector<string>)right);
  for (unsigned int idx = 0; idx < test3.size(); idx++)
    cout << test3[idx] << "\n";


  // Test leftPad
  cout << "== Testing leftPad ==\n";
  char pad = '*';
  vector<string> test4 = string_util.leftPad((const vector<string>)input, pad);
  for (unsigned int idx = 0; idx < test4.size(); idx++)
    cout << test4[idx] << "\n";
}


Output:
== Testing Join ==
The,quick,brown,fox,jumps,over,the,lazy,dog
== Testing reverse ==
dog
lazy
the
over
jumps
fox
brown
quick
The
```

```
== Testing combine ==
Mr.Jones
Mr.Smith
Mr.Williams
Mrs.Jones
Mrs.Smith
Mrs.Williams
== Testing leftPad ==
**The
quick
brown
**fox
jumps
*over
**the
*lazy
**dog
```