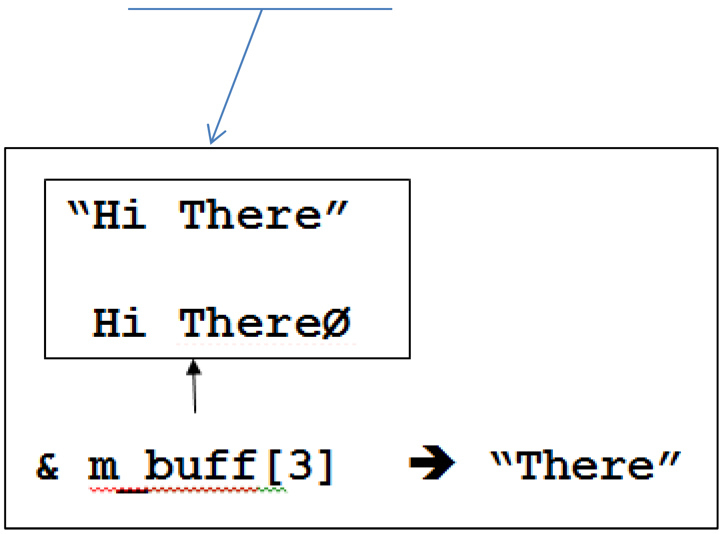
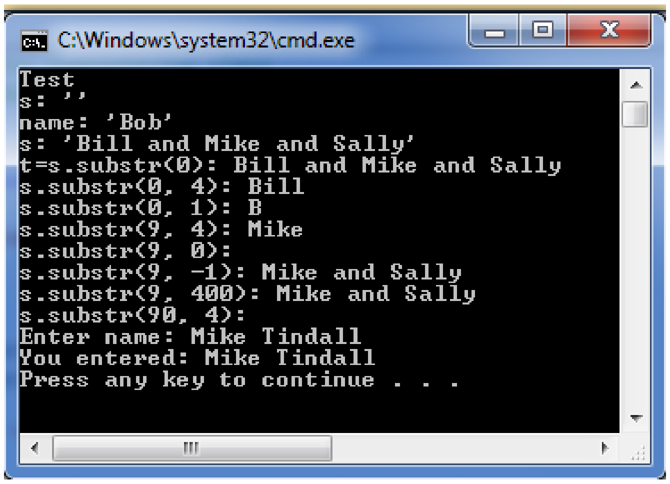
```
// SimpleString.h
// A fixed-size array implementation of a string ADT
// Written by: Mike Tindall, CSC 2430

#ifndef _SIMPLESTRING_H
#define _SIMPLESTRING_H

#include <iostream>
using namespace std;

class SimpleString {
public:
     SimpleString();                        // Default Constructor
     SimpleString(const char s[]);          // Convert Constructor

     int length() const;
     // Current string length

     void concat(const SimpleString &s);
     // Concatenate s to end of string

     int  compare(const SimpleString &s) const;
     // Compare string to s.  Return < 0,  == 0, or  > 0

     char getchar(const int position) const;
     // Retrieve string[position], 0 if out of range

     int  findchar(const char ch) const;
     // Find ch in string.  -1 if not found

     SimpleString substr(const int start, const int len = -1) const;
     // Return new SimpleString with value
     //    that is substring of original

     void readline(istream &in);
     // Read next line into string from input stream in

     const char* toString() const;
     // Return pointer to char[] buffer

private:
     enum { MAX_SIMPLESTRING = 100 };

     // The actual string buffer array
     char m_buff[MAX_SIMPLESTRING];
};

#endif
```

```
// SimpleString.cpp
// Implementation of ADT SimpleString
// Mike Tindall, CSC 2430

#include <iostream>
#include <cstring>
using namespace std;

#include "SimpleString.h"

// Default Constructor
SimpleString::SimpleString()
{
  m_buff[0] = 0;  // empty null-terminated c-string buffer
}

// Convert Constructor
SimpleString::SimpleString(const char s[])
{
  strncpy_s(m_buff, MAX_SIMPLESTRING, s, MAX_SIMPLESTRING-1);
}

int SimpleString::length() const
{
  return(strlen(m_buff));
}

void SimpleString::concat(const SimpleString &s)
{
  strncat_s(m_buff,   MAX_SIMPLESTRING,
            s.m_buff, (MAX_SIMPLESTRING - strlen(m_buff) - 1) );
  return;
}

int SimpleString::compare(const SimpleString &s) const
{
  return(strcmp(m_buff, s.m_buff));
}

char SimpleString::getchar(const int position) const
{
  if( (position < 0) || (position >= length()) )
     return(0);

  return(m_buff[position]);
}
```

```
int SimpleString::findchar(const char ch) const
{
     int len = length();
     for(int i=0; i < len; ++i)
     {
          if(m_buff[i] == ch)
               return(i);
     }

     return(-1);
}

void SimpleString::readline(istream &in)
{
     in.getline(m_buff, MAX_SIMPLESTRING);
}

const char* SimpleString::toString() const  // char *: buff pointer
{
     return(m_buff);
}

SimpleString SimpleString::substr(const int start, const int len) const
{
     // Validate parameters, handle special cases
     if(start < 0 || start >= length())
          return(SimpleString());       // return empty SimpleString

     int sublen = len;
     if(len < 0) sublen = MAX_SIMPLESTRING;

     // strncpy_s( ) copies from starting position to sublen position
     //    or end of m_buff value, whichever comes first.
     char tmp[MAX_SIMPLESTRING];
     strncpy_s(tmp, MAX_SIMPLESTRING, &m_buff[start], sublen);

     return(SimpleString(tmp));
     // return(tmp);
}
```

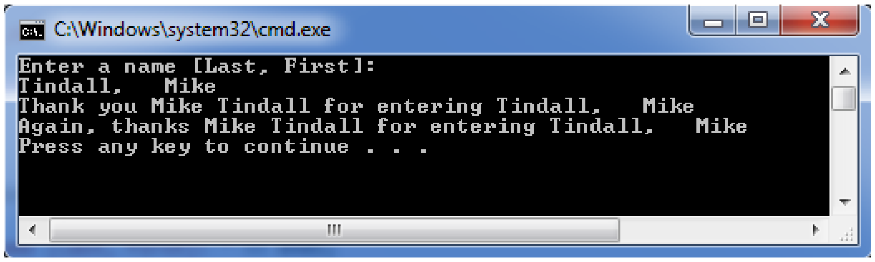




```
Test
s: ''
name: 'Bob'
s: 'Bill and Mike and Sally'
t=s.substr(0): Bill and Mike and Sally
s.substr(0, 4): Bill
s.substr(0, 1): B
s.substr(9, 4): Mike
s.substr(9, 0):
s.substr(9, -1): Mike and Sally
s.substr(9, 400): Mike and Sally
s.substr(90, 4):
Enter name: Mike Tindall
You entered: Mike Tindall
Press any key to continue . . .
```

```
// Test of the SimpleString class
// Mike Tindall, CSC 2430

#include <iostream>
using namespace std;

#include "SimpleString.h"

int main()
{
     SimpleString s;
     SimpleString name("Bob");

     cout << "Test" << endl;

     cout << "s: '" << s.toString() << "'" << endl;

     cout << "name: '" << name.toString() << "'" << endl;

     s = name;

     s = SimpleString("Ryan");

     s = "Bill";

     s.concat(SimpleString(" and Mike"));
     s.concat(" and Sally");

     cout << "s: '" << s.toString() << "'" << endl;

     SimpleString t;
     t = s.substr(0);

     cout << "t=s.substr(0): " << t.toString() << endl;

     cout << "s.substr(0, 4): "   << s.substr(0, 4).toString()   << endl;
     cout << "s.substr(0, 1): "   << s.substr(0, 1).toString()   << endl;
     cout << "s.substr(9, 4): "   << s.substr(9, 4).toString()   << endl;
     cout << "s.substr(9, 0): "   << s.substr(9, 0).toString()   << endl;
     cout << "s.substr(9, -1): "  << s.substr(9, -1).toString()  << endl;
     cout << "s.substr(9, 400): " << s.substr(9, 400).toString() << endl;
     cout << "s.substr(90, 4): "  << s.substr(90, 4).toString()  << endl;

     cout << "Enter name: ";
     name.readline(cin);
     cout << "You entered: " << name.toString() << endl;

}
```

```
// Convert "Last, First" to "First Last" using the SimpleString class
#include <iostream>
using namespace std;

#include "SimpleString.h"
```



```
Enter a name [Last, First]:
Tindall,   Mike
Thank you Mike Tindall for entering Tindall,   Mike
Again, thanks Mike Tindall for entering Tindall,   Mike
Press any key to continue . . .
```

```
///////////////////////////////////////////////////////
// Parameter s is a name in the format  Last, First
// Return a new string value in the format  First Last
SimpleString reverse(const SimpleString &s)
{
      // Find the comma between the names
      int commaloc = s.findchar(',');
      if(commaloc < 0)
            return(s);

      // Find the start of the Firstname
      int firstloc = commaloc + 1;
      while(s.getchar(firstloc) == ' ')
            ++firstloc;

      // Construct the return string value
      SimpleString tmp;
      tmp = s.substr(firstloc);
      tmp.concat(SimpleString(" "));
      tmp.concat(s.substr(0, commaloc));

      return(tmp);
}

int main() {
      SimpleString LastFirst;

      cout << "Enter a name [Last, First]: " << endl;
      LastFirst.readline(cin);

      SimpleString FirstLast;

      FirstLast = reverse(LastFirst);

      cout  << "Thank you "     << FirstLast.toString()
            << " for entering " << LastFirst.toString() << endl;

      // Alternate approach, without using FirstLast
      cout  << "Again, thanks "
            << (reverse(LastFirst)).toString()
            << " for entering "
            << LastFirst.toString()  << endl;

      return(0);
}
```