Spring 2024, ISTM, Purdue-FCU 2+2 ECE Program
Advanced C Programming, Quiz 1

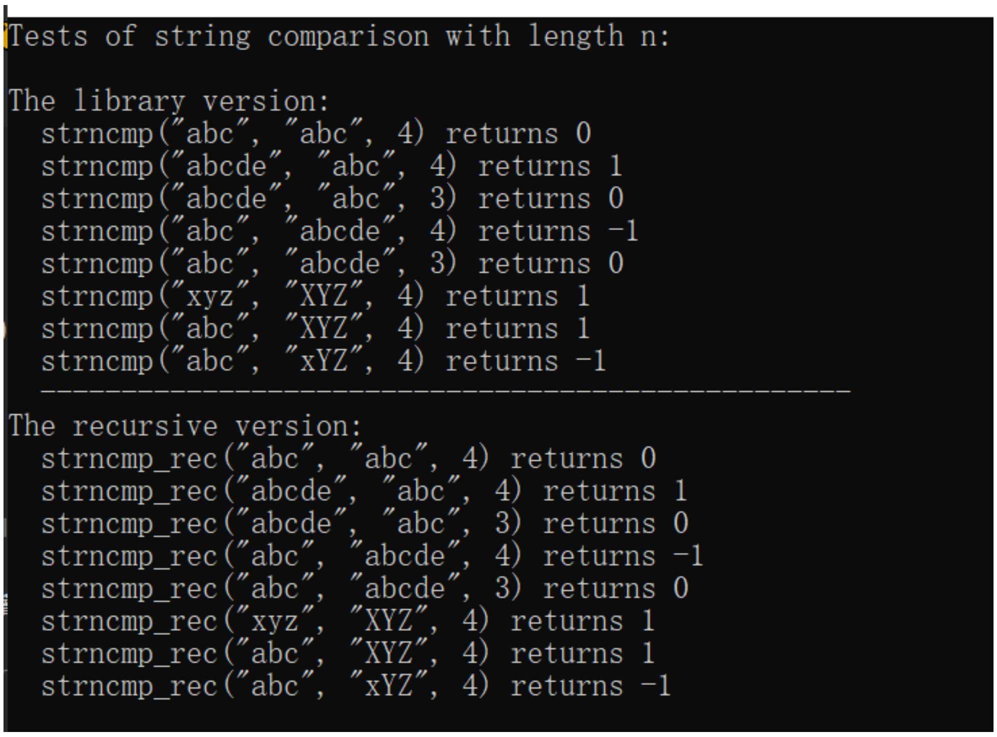
Total TWO FILES for Quiz 1. Use file name quiz1_DXXXXXXX_1.c for Question 1 and file name quiz1_DXXXXXXX_2.c for Question 2, where DXXXXXXX is your student ID. When you finish a question, submit all the above files to the instructor's computer.

1. (40 points) Start with program skeleton **quiz1_skeleton_1.c** and change the file name to **quiz1_DXXXXXXX_1.c**. The following statement is the description of function strncmp() in library <string.h> in C programming language.

   Declaration:
   int strncmp(**const char** *str1, **const char** *str2, size_t n);
   Compares at most the first n bytes of str1 and str2. Stops comparing after the null character. Returns zero if the first n bytes (or null terminated length) of str1 and str2 are equal. Returns less than zero (-1) or greater than zero (1), if str1 is less than or greater than str2, respectively.

   Write a C program to implement and test recursive function
   int strncmp_rec(**const char** *str1, **const char** *str2, size_t n);

   DO NOT use any <string.h> functions in the the implementation of strncmp_rec() and **DO NOT** modify the main program. Program execution example:



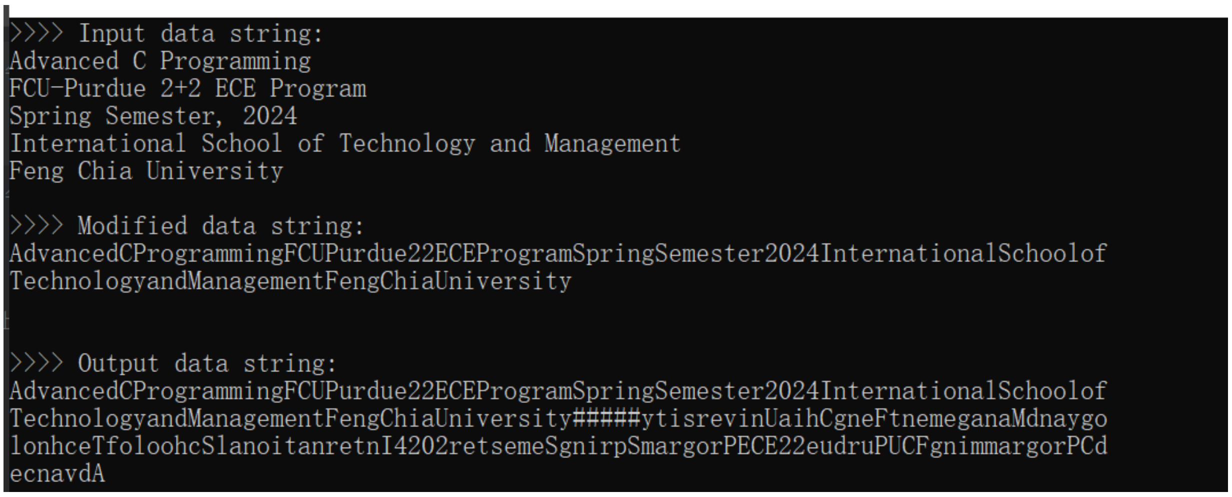
```
Tests of string comparison with length n:

The library version:
  strncmp("abc", "abc", 4) returns 0
  strncmp("abcde", "abc", 4) returns 1
  strncmp("abcde", "abc", 3) returns 0
  strncmp("abc", "abcde", 4) returns -1
  strncmp("abc", "abcde", 3) returns 0
  strncmp("xyz", "XYZ", 4) returns 1
  strncmp("abc", "XYZ", 4) returns 1
  strncmp("abc", "xYZ", 4) returns -1
  --------------------------------------------------
The recursive version:
  strncmp_rec("abc", "abc", 4) returns 0
  strncmp_rec("abcde", "abc", 4) returns 1
  strncmp_rec("abcde", "abc", 3) returns 0
  strncmp_rec("abc", "abcde", 4) returns -1
  strncmp_rec("abc", "abcde", 3) returns 0
  strncmp_rec("xyz", "XYZ", 4) returns 1
  strncmp_rec("abc", "XYZ", 4) returns 1
  strncmp_rec("abc", "xYZ", 4) returns -1
```

(to be continued)

2. (60 points) Start with program skeleton **quiz1_skeleton_2.c** and change the file name to **quiz1_DXXXXXXX_2.c**. Write a C program to perform file operations and string operations as described in the following steps:
   (1) Declare "char *dataIn, *dataOut" to be pointers of input and output data strings, respectively.
   (2) Use file "FCU.txt" as the input testing file.
   (3) Dynamically allocate memory space for dataIn[] to hold the text data of "FCU.txt".
   (4) Read the data string dataIn from file "FCU.txt" character by character using function fgetc() in <stdio.h> until end of file. Print string dataIn.
   (5) Remove all non-alphanumerical characters from string dataIn using functions strpbrk(), strspn(), and strncpy() in <string.h>. Print string dataIn with alphanumerical characters only 80 characters in a line.
   (6) Dynamically allocate memory space for dataOut[] such that string dataOut is the result of concatenating dataIn, "#####", and the reversal string of dataIn. Print string dataOut 80 characters in a line.
   (7) Write the data string dataOut to file "Result.txt" character by character using function fputc() in <stdio.h>.

Program execution example:



```
>>>> Input data string:
Advanced C Programming
FCU-Purdue 2+2 ECE Program
Spring Semester, 2024
International School of Technology and Management
Feng Chia University

>>>> Modified data string:
AdvancedCProgrammingFCUPurdue22ECEProgramSpringSemester2024InternationalSchoolof
TechnologyandManagementFengChiaUniversity


>>>> Output data string:
AdvancedCProgrammingFCUPurdue22ECEProgramSpringSemester2024InternationalSchoolof
TechnologyandManagementFengChiaUniversity#####ytisrevinUaihCgneFtnemeganaMdnaygo
lonhceTfoloohcSlanoitanretnI4202retsemeSgnirpSmargorPECE22eudruPUCFgnimmargorPCd
ecnavdA
```