Develop five modules to replace some of the built-in C/C++ string functions. Use char arrays to store strings. Remember that a null character is used to terminate a string. You will be using loops to process the contents of the array, character by character. Make sure to test your program. **Display the contents of the arrays after each function call.** The strings that are to be passed into the functions should be grabbed from a text file called “in.txt”

The file “in.txt” will contain five lines of data, corresponding to each of the five functions. Here is a sample:

Hello Hi

Great Good

Good bye

Salaam m

Bonjour

**Name of function: void stringCopy(char \*A, char \*B)**

The function will take two strings, A and B, as parameters. It will replace the contents of string A with the contents of string B.

**Name of function: bool stringCompare(char \*A, char \*B)**

This function will take two strings, A and B, as parameters and return a bool. It will return true if the strings are exactly the same, character by character, and false otherwise. The function will be case sensitive, that is, ‘

‘A’ !=’a’.

**Name of function: void stringConcatenation (char \*A, char \*B)**

This function will take two strings, A and B, as parameters. It will combine them such that B is appended to the end of A. For example, given A=”abcd” and B=”efgh”, the function will change A so that it now contains “abcdefgh”. String B will be unchanged by the function.

**Name of function: int stringPosition(char \*A, char B)**

This function will take a string and a character as parameters. It will return the position in the string of the first occurrence of the character. The first character in the string is in position 0. If the string does not contain the character, it returns -1.

**Name of function: int stringLength(char \*A)**

This function will take a char array as a parameter. It will shift all the characters of the string to the right by one and store the length of the string in position zero. The length of the string does not include the null character. The function will return the length which is stored in position zero.

Example:

Array before the function call

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index of array | 0 | 1 | 2 | 3 |
| Value at index | ‘A’ | ‘B’ | ‘C’ | ‘\0’ |

Array after the function call

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index of array | 0 | 1 | 2 | 3 | 4 |
| Value at index | 3 | ‘A’ | ‘B’ | ‘C’ | ‘\0’ |

You **MUST NOT** use any C/C++ built-in string manipulation functions in your implementation!

**File read/write example**

#include <iostream>

#include <fstream>

using namespace std;

//Place these at the global level

ifstream in("input.txt"); //input.txt is the file that we read from

ofstream out("output.txt"); //output.txt is the file that we write into

//This function looks ahead to get rid of all extra characters, such as new lines, spaces, tabs

//and single line comments.

int clean(){

string comments; //store the comments in this variable

while (in.peek()=='\n' || in.peek()=='/' || in.peek()==' ' || in.peek()=='\t'){

if (in.peek()=='/' )

getline(in, comments); //read the entire line because it is a comment

else

in.get(); //eat the newline, space and tab characters

}

return 1;

}

int main(){

char from[10], to[10];

double distance;

// can use in.good() to check if for end of file

while (clean() && in.good()){

in >>from>> to >> distance;

out<<from<<" "<<to<<" "<< distance <<endl;

}

in.close();

out.close();

}

**Sample contents of Input.txt**

//distance between Roseville and Sacramento

Roseville Sacramento 7

//distance between Roseville and Davis

Roseville Davis 30