String Functions in C++ (cstring library)

The 'cstring' library defines various string functions that can be used to perform various operations on strings. This document contains some of the string functions that are mostly used in the programs.

Declaring a character array:
A character array can be declared as:
```
char str[10];
```
This means that this is a character array i.e. a string with 10 characters. Please note that this can also be represented as
```
char *str;
```

The following example is used for all the functions given below:
```
char str[30];
gets(str);
```
Assume that "**`C++ string functions`**" is given as input and str contains this string.

`#include<cstring>` needs to be written to use any of the functions menioned below.

No.	Function	Description
1	strlen (const char *str);	This function returns the length of the string str. The length of the string is the number of characters in the string without the terminating character '\0'. e.g. cout<<"The length of the string: "<<strlen(str); **Output:** The length of the string: **20**
2	char* strcpy (char* dest, char* src);	This is a string copy function. The first parameter is the destination string and the second parameter is the source string. The function strcpy will copy the contents of source string to the destination string, including the terminating null-character '\0'. e.g `char dest[30];` `strcpy(dest, str);` `cout<< "Destination String: "<<dest;` **Output:** Destination String: **C++ string functions**
3	char* strcat (char* dest, const char* src);	This is a string concatenation function. The first parameter is the destination string and the second parameter is the source string. The function '**`strcat()`**' will concatenate/append the contents of source array to the destination array. The terminating null-character of the destination array is overwritten by the first character of the source array and a null-character is introduced in the destination string at the end of new string. e.g `char dest[50]="Hello";` `strcat(dest,str);` `cout<< "Destination String: "<<dest;` **Output:** Destination String: **HelloC++ string functions**

4	int strcmp (char * str1, char * str2);	This function compares two strings. The first parameter is string 1 and the second parameter is string 2 which will be compared by this function. It starts by comparing the first character of both the strings. It compares till it finds a non matching character or the terminating null-character. It returns an integral value. The following is returned by this function. **Return Value Description** **0** States that both the strings are equal **< 0** States that the non-matching character encountered has a lower value in str1 than str2 **> 0** States that the non-matching character encountered has a greater value in str1 than str2 **e.g 1**: `char str1[40]="C++ string functions";` `char str2[40]="C++ string functions";` `cout<<"Result: "<<strcmp(str1,str2);` Output: Result: **0** *The result is 0 as both the strings are equal.* **e.g. 2**: `char str1[40]="C++ functions";` `char str2[40]="C++ string functions";` `cout<<"Result: "<<strcmp(str1,str2);` Output: Result: **-1** *The result is -1 as the non matching character 'f' < 's'* **e.g. 3**: `char str1[40]="C++ string functions";` `char str2[40]="C++ functions";` `cout<<"Result: "<<strcmp(str1,str2);` Output: Result: **1** *The result is 1 as the non matching character 's' > 'f'*

Program using C++ string functions

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

void mergesort(char A[100][101], int start, int end);
void mergeSortedSubarrays(char A[100][101], int start, int mid,int end);
bool lexEarlier(char s1[], char s2[]);

int main()
{
  int n;
  char A[100][101];
  cout << "Give number of strings (between 1 and 100): ";
  cin >> n;
  if ((n <= 0) || (n > 100)) {
    cout << "Invalid value of n: " << n << endl;
    return -1;
  }
```

```
  cout << "Give " << n << " strings (separated by newlines):" << endl;
  cout << "Each string should be at most 100 characters long." << endl;

  // Reading strings using a larger temporary char array
  // Just to check if the user provided string has <= 100 characters

  char nextInput[201];
  for (int i = 0; i < n; ) {
    cout << "String " << i+1 << ": ";
    cin >> nextInput;
    if (strlen(nextInput) > 100) {
      cout << "Your input exceeded 100 characters -- exiting!" << endl;
      return -1;
    }
    strcpy(A[i++], nextInput);
  }

  cout << "+++++++++++++++++" << endl;
  cout << "Unsorted array: " << endl;
  cout << "+++++++++++++++++" << endl;
  for (int i = 0; i < n; i++) {
    cout << A[i] << endl;
  }

  mergesort(A, 0, n);

  cout << "+++++++++++++++++" << endl;
  cout << "Sorted array: " << endl;
  cout << "+++++++++++++++++" << endl;
  for (int i = 0; i < n; i++) {
    cout << A[i] << endl;
  }

  return 0;
}

void mergesort(char A[100][101], int start, int end)
{
  if (end == start + 1) {return;}
  int mid = (start + end)/2;
  mergesort(A, start, mid);
  mergesort(A, mid, end);
  mergeSortedSubarrays(A, start, mid, end);
  return;
}

void mergeSortedSubarrays(char A[100][101], int start, int mid, int end)
{
  int i, j, index = start;
  char tempA[100][101];

  for (i=start, j=mid; ((i < mid) || (j < end)); ) {
    if ((i < mid) && (j < end)) {
      if (lexEarlier(A[j], A[i])) {strcpy(tempA[index], A[j++]);}
      else {strcpy(tempA[index], A[i++]);}
    }
    else {
```

```
      if (i < mid) {strcpy(tempA[index], A[i++]);}
      else {strcpy(tempA[index], A[j++]);}
    }
    index++;
  }

  for (int i = start; i < end; i++) {
    strcpy(A[i], tempA[i]);
  }

  return;
}

bool lexEarlier(char s1[], char s2[])
{
  if (strcmp(s1, s2) < 0) {
    return true;
  }
  else {
    return false;
  }
}
```