String Assignment

Notes:

This Assignment needs the application of concepts of pointers, arrays, dynamic memory allocation, operators, and control flow constructs.

- Handle common string manipulation errors. Such as unbounded string copies, off-by-one errors and null-termination errors
- Write functions wherever necessary.
- All programs should follow ***C-11 standard*** and ***GESL coding guidelines.***
- Use ***-Wall*** and ***-Wstrict-prototypes*** options while compiling to arrest any warnings.

1. Write a function:
   char *strcopy(char *****dest**, const char *****src*****);*** that copies the string pointed to by *src*, including the terminating null byte ('\0'), to the buffer pointed to by *dest* and returns a pointer to the destination string *dest*.
2. Write a function:
   char *strncopy(char *****dest**, const char *****src*****, int*** *n****);*** that copies at most n bytes of *src* into *dest*. If there is no null byte among the first *n* bytes of *src*, the string placed in *dest* will not be null terminated. If the length of *src* is less than *n, strncopy*() writes additional null bytes to *dest* to ensure that a total of *n* bytes are written. The function returns a pointer to the destination string *dest*.
3. Write a function:
   int strindex(char *****str**, char*** *c****);*** that returns the index of the first occurrence of character '*c*' in the given string *str.* For example, if *str* = "Global Edge", *c* = 'l' The function returns index 1 as the output.
4. Write a function:
   char *strappend(char *****dest**, const char *****src*****);*** that appends the *src* string to the *dest* string, overwriting the terminating null byte ('\0') at the end of *dest*, and then adds a terminating null byte. The function returns a pointer to the resulting string *dest*.
5. Write a function:
   char *strnappend(char *****dest**, const char *****src*****, int*** *n* ***);*** that appends at most *n* characters from *src* to the *dest*. The *src* does not need to be null terminated if it contains *n* or more bytes. The resulting string in *dest* is always null terminated and function returns a pointer to the resulting string *dest*.
6. Write a function:
   int strcomp(const char *****s1**, const char *****s2*****);*** to compare two strings *s1* and *s2* and returns:
   a. 0 if *s1* is equal to *s2*
   b. 1 if *s1* is greater than *s2*
   c. -1 if *s1* is less than *s2*
7. Write a function:
   int strcompcase(const char *****s1**, const char *****s2*****);*** to compare two strings *s1* and *s2* ignoring the cases and returns:
   a. 0 if *s1* is equal to *s2*
   b. 1 if *s1* is greater than *s2*
   c. -1 if *s1* is less than *s2*
8. Write a function:
   int strspan(const char *****s1**, const char *****s2*** ***);*** that calculates the length (in bytes) of the initial segment of *s1* which consists entirely of bytes in *s2*. The function returns the count.

For example, if *s1* = "globaledge", *s2* = "learning" then 1st char 'g' in *s1* is found in *s2*, 2nd char 'l' in *s1* is found in *s2* but 3rd char 'o' in s1 is not found in *s2*. So, the initial segment of 2 characters in *s1* is found in *s2*. The count is 2 and same is returned by the function.

9. Write a function:
   char *strtoken(char *str, const char *delim); that parses *str* until it encounters any of the delimiters in *delim*. For more info, read man page of strtok.
   For example, if *str* = "cd ../c/experiments/string", *delim* = "/" then
   1st call : token = strtoken(*str*, *delim*) //results in token = cd ..
   Subsequent 2nd call : token = strtoken(*NULL*, *delim*) //results in token = c
   Subsequent 3rd call : token = strtoken(*NULL*, *delim*) //results in token = experiments
10. Write a function:
    int strpalin(char *str); that checks whether *str* is palindrome or not. The function returns 1 if palindrome else 0.
11. Write a function:
    char *strrev(char *str); that reverses the string *str*
    and returns the result.
    For example, if *str* = "GlobalEdge", the output *str* = "egdElabolG"
12. Write a function:
    char *strsqueeze(char *str); that squeezes the consecutive similar characters in *str*
    and returns the modified *str*. For Example, if *str* = "Hellooo", the output *str* = "Helo"
13. Write a function:
    int *strrot(const char *str, const char *rstr);
    that checks whether *rstr* is rotated (left or right) string of *str* or not. The function returns 1 if true else 0. For Example,
    if *str* = "Global", *rstr* = "balGlo", the output is 1, as left rotation of *str* by 3 results in *rstr* if
    str = "Global", *rstr* = "alGlob", the output is 1, as right rotation of *str* by 2 results in *rstr* if
    str = "Global", *rstr* = "balolG", the output is 0, as rstr not rotated string of *str*
14. Write a function:
    char *strrem(char *str, const char *sstr); that checks for the substring *sstr* in *str* and if found removes the substring *sstr* in *str* and returns the modified string *str*. If the substring *sstr* is not found, it returns the original string *str*.
    For example, if *str* = "GlobalEdgeSoftwareLtd", *sstr* = "Edge" then output
    str = "GlobalSoftwareLtd"
15. Write a function:
    char *strinschr(char *str, const char c, int pos); that will insert character *c* in *str* at given *pos.* The function returns modified string *str*.
    For example, if *str* = "GlobalEdge", *c* = ' '(space), *pos* = 6
    The output *str* = "Global Edge"