PROBLEMS

# 1.

Write a function that will find how many times a given character occurs in given string. Example

For the string HELLO FINKI character L occurs 2 times.

# 2.

Write a function that will return the length of a string. Also write a recursive solution.

# 3.

Write a program that will print substring from given string, determined with the position and the length as parameters read from SI. The substring starts from the character on the position counted from left to right.

# 4.

Write a function that will check if one string is substring of some other string.

# 5.

Write a function that will check if given string is palindrome. Palindrome is a string that is read same from left to right and from right to left.

# 6.

Write a function that will check if given sentence is a palindrome. Ignore the empty spaces, punctuations characters and the case of letters.

# 7.

Write a function that for a given string will if it’s complex enough to become a password. Every password must have at least eight characters, one uppercase letter, one digit and one special character.

# 8.

Write a function that for will change the case of the letters and will remove all digits and special characters.

# 9.

Write a function that will trim a string (remove blanks at front and end of string).

SOLUTIONS

# 1.

*/\*Write a function that will find how many times a given character occurs in given string. Example  
For the string HELLO FINKI character L occurs 2 times.  
\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** countOccurrences(**char** \*str,**char** c){  
 **int** count=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(tolower(str[i])== tolower(c)){  
 count++;  
 }  
 }  
 **return** count;  
}  
  
**int** main(){  
 **char** c;  
 scanf(**"%c\n"**,&c);  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 printf(**"%d"**, countOccurrences(str,c));  
  
 **return** 0;  
}

# 2.

*/\*Write a function that will return the length of a string. Also write a recursive solution.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** lenOfStr(**char** \*str){  
 **int** length=0;  
 **while**(\*str!=**'\0'**){  
 length++;  
 str++;  
 }  
 **return** length;  
}  
  
**int** lenOfStrRecursive(**char** \*str){  
 **if**(\*str==**'\0'**){  
 **return** 0;  
 } **else**{  
 **return** 1+ lenOfStrRecursive(str+1);  
 }  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 printf(**"%d"**,lenOfStrRecursive(str));  
 **return** 0;  
}

# 3.

*/\*Write a program that will print substring from given string, determined with the position and the length as parameters  
read from SI. The substring starts from the character on the position counted from left to right.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** subStr(**char** \*str, **int** start, **int** end){  
 **for**(**int** i=start ; i<start+end && i< strlen(str) ; i++){  
 printf(**"%c"**,\*(str+i));  
 }  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 **int** start,length;  
 scanf(**"%d%d"**,&start,&length);  
 subStr(str,start,length);  
 **return** 0;  
}

# 4.

*/\*Write a function that will check if one string is substring of some other string.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** isSubStr(**char** \*smaller, **char** \*bigger){  
 **int** lenSmaller= strlen(smaller);  
 **int** lenBigger= strlen(bigger);  
 **if**(lenBigger<lenSmaller){  
 **return** 0;  
 }  
 **char** \*result= strstr(bigger,smaller);  
 **return** \*result!=**'\0'**;  
}  
  
**int** main(){  
 **char** smaller[100];  
 fgets(smaller, **sizeof** (smaller), **stdin**);  
 smaller[strlen(smaller) - 1]=**'\0'**;  
  
 **char** bigger[100];  
 fgets(bigger, **sizeof** (bigger), **stdin**);  
 bigger[strlen(bigger) - 1]=**'\0'**;  
  
 **if**(isSubStr(smaller, bigger)){  
 printf(**"YES"**);  
 }**else**{  
 printf(**"NO"**);  
 }  
 **return** 0;  
}

# 5.

*/\*Write a function that will check if given string is palindrome. Palindrome is a string  
that is read same from left to right and from right to left.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** isPalindrome(**char** \*str){  
 **for**(**int** i=0,j= strlen(str)-1 ; i< strlen(str) ; i++,j--){  
 **if**(str[i]!= str[j]){  
 **return** 0;  
 }  
 }  
 **return** 1;  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 **if**(isPalindrome(str)){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **return** 0;  
}

# 6.

*/\*Write a function that will check if given sentence is a palindrome. Ignore the empty spaces, punctuations characters and the case of letters.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** clearString(**char** \*str){  
 **int** counter=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i])){  
 str[counter]= str[i];  
 counter++;  
 }  
 }  
 str[counter]=**'\0'**;  
}  
  
**int** isPalindrome(**char** \*str){  
 **for**(**int** i=0,j= strlen(str)-1 ; i< strlen(str) ; i++,j--){  
 **if**(tolower(str[i])!= tolower(str[j])){  
 **return** 0;  
 }  
 }  
 **return** 1;  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 clearString(str);  
 **if**(isPalindrome(str)){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **return** 0;  
}

# 7.

*/\*Write a function that for a given string will if it’s complex enough to become a password.  
Every password must have at least eight characters, one uppercase letter, one digit and one special character.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** passwordCheck(**char** \*str){  
 **if**(strlen(str)<8){  
 **return** 0;  
 }  
 **int** digit=0,specialCharacter=0,upperCase=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isdigit(str[i])){  
 digit=1;  
 }  
 **if**(!isalnum(str[i])){  
 specialCharacter=1;  
 }  
 **if**(isupper(str[i])){  
 upperCase=1;  
 }  
 }  
 **return** digit==1 && specialCharacter==1 && upperCase==1;  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 **if**(passwordCheck(str)){  
 printf(**"Strong password"**);  
 } **else**{  
 printf(**"Weak password"**);  
 }  
 **return** 0;  
}

# 8.

*/\*Write a function that for will change the case of the letters and will remove all digits  
and special characters.\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** clearStr(**char** \*str){  
 **int** counter=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(isalpha(str[i])){  
 **if**(isupper(str[i])){  
 str[counter]= tolower(str[i]);  
 } **else if**(islower(str[i])){  
 str[counter]= toupper(str[i]);  
 }  
 counter++;  
 }  
 }  
 str[counter]=**'\0'**;  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 clearStr(str);  
 puts(str);  
 **return** 0;  
}

# 9.

*/\*Write a function that will trim a string (remove blanks at front and end of string).\*/*#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
void** clearStr(**char** \*str){  
 **int** i=0;  
 **while**(!isalpha(str[i])){  
 i++;  
 }  
 **for**(**int** j=0 ; j< strlen(str) ; j++,i++){  
 str[j]=str[i];  
 }  
 **int** k= strlen(str);  
 **while**(!isalpha(str[k])){  
 k--;  
 }  
 str[k+1]=**'\0'**;  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 clearStr(str);  
 puts(str);  
 **return** 0;  
}