Homework: C Characters and Strings

This document defines the homework assignments from [the "C Programming" Course @ Software University]. Please submit as homework a single **zip** / **rar** / **7z** archive holding the solutions (source code) of all below described problems.

Problem 1. Reverse String

Write a program that reads a string from the console, reverses it and prints the result back at the console.

Input	Output
sample	elpmas
24tvcoi92	29iocvt42

Problem 2. String Length

Write a program that reads from the console a string of **maximum 20 characters**. If the length of the string is less than 20, the rest of the characters should be filled with asterisks '**'. Print the resulting string on the console.

Input	Output
Welcome to SoftUni!	Welcome to SoftUni!*
a regular expression (abbreviated regex or regexp and sometimes called a rational expression) is a sequence of characters that forms a search pattern	a regular expression
C#	C#******************

Problem 3. Series of Letters

Write a program that reads a string from the console and replaces all series of consecutive identical letters with a single one.

Input	Output
aaaaabbbbbcdddeeeedssaa	abcdedsa

Problem 4. Count Substring Occurrences

Write a program to **find how many times a given string appears in a given text as substring**. The text is given at the first input line. The search string is given at the second input line. The output is an integer number. Please ignore the **character casing**. **Overlapping** between occurrences is **allowed**. Examples:

Input	Output
Welcome to the Software University (SoftUni)! **Wel**come to programming. Programming is **wel**lness for developers, said Max**well**. wel	4
aaaaaa aa	5
ababa caba aba	3

Welcome to SoftUni Java	0

Problem 5. Text Filter

Write a program that takes a **text** and a **string of banned words**. All words included in the ban list should be replaced with **asterisks** "*", equal to the word's length. The entries in the ban list will be separated by a **comma** and **space** ", ".

The ban list should be entered on the first input line and the text on the second input line. Example:

Input	Output
Linux, Windows I'd just like to interject for a moment. What you're referring to as **Linux**, is in fact, GNU/**Linux**, or as I've recently taken to calling it, GNU plus **Linux** Sincerely, a **Windows** user	I'd just like to interject for a moment. What you're referring to as *****, is in fact, GNU/*****, or as I've recently taken to calling it, GNU plus ***** Sincerely, a ******* user

Problem 6. Palindromes

Write a program that extracts from a given text all palindromes, e.g. ABBA, lamal, exe and prints them on the console on a single line, separated by comma and space. Use spaces, commas, dots, question marks and exclamation marks as word delimiters. Print only **unique** palindromes, **sorted** lexicographically. String comparison should be **case-sensitive** as shown in the example below.

Input	Output
Hi,exe? ABBA! Hog fully a string. Bob	a, ABBA, exe

Problem 7. Implement a String Copy Function

The standard C function **strncpy(char *dest, const char *src, size_t n)** copies a chunk from the src string to the dest buffer. Try implementing that function manually. Try doing it without a buffer, by returning a string created in the body of the function. Does it work? Think why it does or it doesn't. How can you modify it to work/break it.

Input	Output
strncpy(buffer, "SoftUni", 7)	SoftUni
strncpy(buffer, "SoftUni", 3)	Sof
strncpy(buffer, "C is cool", 0)	(empty string)
char* result = strncpy("SoftUni", 7)	???

Problem 8. Implement a String Concatenation Function

The standard C function **strncat(char *dest, const char *src, size_t n)** concatenates the strings located in the **src** and the **dest** buffer into the **dest** buffer. The variable **n** shows the length from the **src** string to be concatenated. Try implementing that function yourself. Think about how you can get the length of the two strings.

Input	Output
char buffer[10] = "Soft"; strncat(buffer, "Uni", 7)	SoftUni

`char buffer[5] = "Soft";` `strncat(buffer, "ware University", 15)`	`Segmentation Fault. Think why`
`char buffer[10] = "C";` `strncat(buffer, " is cool", 8)`	`C is cool`
`char * buffer = "C";` `strncat(buffer, " is cool", 8)`	`Segmentation Fault. Think why`

Problem 9. Implement a Word Count function

Implement a function which counts the words in a given input. The function accepts as parameter the input from which to count the words and the delimiter separating the words. The declaration could be as follows:

```
int wc(char * input, char delimiter);
```

Input	Output
`wc("Hard Rock, Hallelujah!", ' ');`	`3`
`wc("Hard Rock, Hallelujah!", ',');`	`2`
`wc("Uncle Sam wants you Man", ' ');`	`5`
`wc("Beat the beat!", '!');`	`2 (An empty string counts as a word as well)`

Problem 10. Implement a String Length function

The standard C function **`size_t strlen(const char *s)`** returns the size of the input string. Try implementing it yourself. Think about how strings are represented in C in order to calculate their length.

Input	Output
`strlen("Soft");`	`4`
`strlen("SoftUni");`	`7`

`char buffer[10] =`
`{ 'C', '\0', 'B', 'a', 'b', 'y' };`
 	
`strlen(buffer);`
	`1`

`char buffer[] =`
`{ 'D', 'e', 'r', 'p', '\0' };`
 	
`strlen(buffer);`
	`4`

Problem 11. Implement a String Search function

Implement a function which checks whether a string appears as a substring in another string. It should return 1 if the string occurs and 0 if it does not. Its declaration could be:

```
int strsearch(char * src, char * substr);
```

Input	Output
`strsearch("SoftUni", "Soft");`	`1`
`strsearch("Hard Rock", "Rock");`	`1`
`strsearch("Ananas", "nasa");`	`0`

`strsearch("Coolness", "cool");`	`0`

Problem 12. Implement a Substring function

Implement a function which extracts a substring from a given string by specifying the position from which to extract and the length to extract. The declaration could be as follows:

int substr(char * src, int position, int length);

Input	Output
`substr("Breaking Bad", 0, 2);`	`Br`
`substr("Maniac", 3, 3);`	`iac`
`substr("Maniac", 3, 5);`	`iac`
`substr("Master Yoda", 13, 5);`	`(empty string)`

Problem 13. Read Line Function

Write a function that **reads an entire line** from the standard input stream (until end of line ('**\n**') or end of file (**EOF**) and returns a pointer to the **read string**. The function should be able to read lines of **unknown size**.

The returned string's **length** should be equal to its **allocated memory**.

Problem 14. Matrix of Palindromes

Write a program to generate the following matrix of palindromes of **3** letters with **r** rows and **c** columns:

Input	Output

`3 6`	`aaa aba aca ada aea afa`
	`bbb bcb bdb beb bfb bgb`
	`ccc cec cdc cfc cgc chc`

`2 3`	`aaa aba aca` `bbb bcb bdb`
`1 1`	`aaa`
`1 3`	`aaa aba aca`

Problem 15. Remove Names

Write a program that takes as input two lists of names and **removes from the first list all names given in the second list**. The input and output lists are given as words, separated by a space, each list at a separate line. Examples:

Input	Output
`Peter Alex Maria Todor Steve Diana Steve` `Todor Steve Nakov`	`Peter Alex Maria Diana`
`Hristo Hristo Nakov Nakov Petya` `Nakov Vanessa Maria`	`Hristo Hristo Petya`

Problem 16. Longest Word in a Text

Write a program to find the longest word in a text. Examples:

Input	Output
`Welcome to the Software University.`	`University`
`The C# Basics course is awesome start in programming with C# and Visual Studio.`	`programming`

Problem 17. XML Parser

Write a program that reads n lines in XML format and parses their contents. A line is considered valid if it follows the format <**{tag}**>**{data}**</**{tag}**>. In case of invalid line format, print "**Invalid format**".

Input	Output

`<name>Gosho</name>`
`<age>13</age>`
`<eye-color>blue</eye-color>`
	`Name: Gosho`
	`Age: 13`
	`Eye-color: blue`

`<nGosho</name>`
`<sex>male</sex>`
`<height>176cm<height>`
	`Invalid format`
	`Sex: male`
	`Invalid format`

Problem 18. Longest Area in Array

Write a program to find the **longest area of equal elements** in array of strings. You first should read an integer **n** and **n** strings (each at a separate line), then find and print the longest sequence of equal elements (first its length, then its elements). If multiple sequences have the same maximal length, print the **leftmost** of them. Examples:

Input	Output
6 hi hi hello ok ok ok	3 ok ok ok
2 SoftUni Hello	1 SoftUni
4 hi hi hi hi	4 hi hi hi hi
5 wow hi hi ok ok	2 hi hi

Problem 19.* Sort City Names

You are given a list of cities. You have to process them and sort them in ascending order. On the first input line, you are given the count of the cities. Use an algorithm other than bubble sort.

Input	Output
4 Sofia Burgas Aitos Pleven	Aitos Burgas Pleven Sofia
2 Svoge Pamporovo	Pamporovo Svoge
1 New york	New york

Problem 20.*Print City Matrix Diagonal

You are given a square matrix of city names. Your task is to print the names of those cities which are stationed on the matrix's main diagonal. On the first input line you are given the count of the rows. The names of the cities will consist of one word only.

Input	Output
4 **Sofia** Burgas Aitos Pleven Varna **Skopie** Athens Burkley London Plovdiv **Svishtov** Ohrid Paris Vienna Berlin **Manchester**	Sofia Skopie Svishtov Manchester
3 **Moscow** Brussels Luxemburg Varna **Madrid** Lissabon Munchen Copenhagen **Pleven**	Moscow Madrid Pleven
1 **Yorkshire**	Yorkshire

Problem 21.*Diamond

You are given as input a number which represents the height and width of a diamond. Your task is to draw it according to the given metrics. See the examples for clarification.

Input	Output
5	..*.. .*.*. *...* .*.*. ..*..

7	...*... ..*.*.. .*...*. *.....* .*...*. ..*.*.. ...*...
3	.*. *.* .*.

Problem 22.* Letters Change Numbers

This problem is from the Java Basics exam (8 February 2015). You may check your solution here.

Nakov likes Math. But he also likes the English alphabet a lot. He invented a game with numbers and letters from the **English** alphabet. The game was simple. You get a string consisting of a **number between two letters**. Depending on whether the letter was in front of the number or after it you would perform different mathematical operations on the number to achieve the result.

First you start with the letter **before** the number. If it's **Uppercase** you **divide** the number by the letter's **position** in the alphabet. If it's **lowercase** you **multiply** the number with the letter's position. **Then** you move to the **letter after** the number. If it's **Uppercase** you **subtract** its position from the resulted number. If it's **lowercase** you **add** its position to the resulted number. But the game became too easy for Nakov really quick. He decided to complicate it a bit by doing the same but with **multiple** strings keeping track of only the **total sum** of all results. Once he started to solve this with more strings and bigger numbers it became quite hard to do it only in his mind. So he kindly asks you to write a program that **calculates the sum of all numbers after the operations on each number have been done**.

For example, you are given the sequence "**A12b s17G**". We have two strings – **"A12b"** and **"s17G"**. We do the operations on each and sum them. We start with the letter before the number on the first string. **A is Uppercase** and its position in the alphabet is **1**. So we divide the number 12 with the position 1 (**12/1 = 12)**. Then we move to the letter after the number. **b is lowercase** and its position is 2. So we add 2 to the resulted number (**12+2=14)**. Similarly for the second string **s is lowercase** and its position is 19 so we multiply it with the number (**17*19 = 323)**. Then we have Uppercase G with position 7, so we subtract it from the resulted number (**323 – 7 = 316)**. Finally we sum the 2 results and we get **14 + 316=330**;

Input

The input comes from the console as a **single line, holding the sequence of strings**. Strings are separated by **one or more white spaces**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print at the console a single number: the **total sum of all processed numbers** rounded up to **two digits** after the decimal separator.

Constraints

- The **count** of the strings will be in the range **[1 … 10].**
- The numbers between the letters will be integers in range **[1 … 2 147 483 647].**
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output	Comment
A12b s17G	330.00	12/1=12, 12+2=14, 17*19=323, 323-7=316, **14+316=330**
P34562Z q2576f H456z	46015.13	
a1A	0.00	