KADI: http://46.101.216.112

CSE102#hw06

Due Date: April 8, 23:55

In this homework you will write programs to exercise with strings, loops, arrays, functions and pointers. Please do not forget that your program should work exactly as the example runs given below the part definitions. Submit your homework via KADI.

#part1

In this part of the homework, you will calculate the character frequency of a given string. You will handle only the ASCII letters and digits. Any other characters such as punctuation marks and whitespaces will be handled as the same.

Second parameter's length is fixed and equals to 37. First 26 elements of it are letters in ascending order, next 10 elements are digits and the last one is for the other characters. You should not modify the order. Note that case sensitivity has no effect on letter frequency.

Signature

```
void freq(char* str, int* fr);
```

Sample Usage

```
int fr[37];
freq("abCd Ef ghIj kLM nopr stuv yzwxq 123 4 5 6 7 890 00 ?? |",fr);
```

fr's value after function call

```
A => 1
B => 1
C => 1
D => 1
E => 1
F => 1
G => 1
H => 1
I => 1
J => 1
K => 1
L => 1
M => 1
N => 1
O => 1
P => 1
Q => 1
R => 1
S => 1
T => 1
U => 1
V => 1
W => 1
X => 1
Y => 1
Z => 1
0 => 3
1 => 1
2 => 1
3 => 1
4 => 1
5 => 1
6 => 1
7 => 1
8 => 1
9 => 1
Others => 18
```

Note: left-hand side values are for convenience, do not print anything in function.

#part2

In this part, you will write a function for finding substring(s) in a given string. Instead of matching the exact substring, you are asked to match only odd indexed characters. For example, if we look for the “lerom” in “lorem ipsum dolor sit amet.” you’ll match the first word because 1st, 3rd and 5th letters of the “lorem” matches with “lerom”.

The return value of this function is a character pointer to the first letter of the first occurrence of the needle in the haystack. If there is no match, function returns null (i.e. 0).

Signature
`char* matcher(char* haystack, char* needle);`
Sample Usage
`char* ptr = matcher(“lorem ipsum dolor sit amet.”, “lerom”);`
ptr’s value after function call
“lorem ipsum dolor sit amet.” ↑

#part3

In this part, you will implement a function that counts the occurrences of a substring in the a given string using the matcher function that you wrote in part2. Note that you will match only the odd indexed characters.

Signature
`int count(char* str, char* substr);`
Sample Usage
`count("ecececece","ece");`
Return value
This call will return with the value of **3**. ecececece ↑ ↑ ↑

#notes

- You should not send your main function. You should send the functions described above and your helper functions if you have any.
- Use the structures you learned such as loops, conditions and functions whenever needed. The usage of these structures is as important as the correct execution of your program.
- The assignment must be your original work. Duplicate or very similar assignments are both going to be considered as cheating.
- Ask your questions via moodle.