Sentence Manipulator

(Functions ,Arrays and Strings)

Write a program that manipulates one sentence. A sentence is a sequence of words over an 'A'-'Z' alphabet. A sentence contains at most 30 characters (spaces are not counted; number of spaces between words *in input sentence* is not important). In case the input contains symbols different from 'A'-'Z' or space, ask user to type data again after following massage:

ERROR: Incorrect data, try again.

A sentence will be held as a single array of characters so that distance between words in the same array is **exactly one space**.
The program displays a user menu, reads one out of 10 options (numbers 1 – 10 only) prints the result of suitable operation and goes back to the display menu.

The menu is :

******** Main menu ********

1 : Input the sentence
2 : Print the sentence
3: Reverse the sentence
4 : Rotate left N
5 : Rotate right N
6 : Sort the sentence
7 : Search the word
8 : Count number of words
9 : Display substring M
10 :Quit

****

Please, enter your choice >

The program runs until the user chooses the option 10 (Quit).
In case there is an error in the command, the program asks the user to try command again after following message:

ERROR: Incorrect command, try again.

In case the sentence isn't initialized and the user choosed commands 2-9 the following message will appear:

ERROR: The sentence is empty.

and the program goes back displaying the menu.

Commands Descriptions:

1. Input the sentence

Description: Reads a sentence , checks it and displays exactly as it was entered.
Number of spaces between words in input sentence is not important.
If the sentence is valid it will be held as array of characters.
NOTE: A distance between words in the same array is **exactly one space.**

Example1:

Please, enter your choice > 1

Please, enter your sentence > HAVE A NICE DAY

HAVE A NICE DAY

Example2:

Please, enter your choice >1

Please, enter your sentence > DOGS LIKE CATS

DOGS LIKE CATS

Example3:

Please, enter your choice >1

Please, enter your sentence >HAVE A NICE DAY !

ERROR: Incorrect data, try again.

2. Print the sentence

Description: Prints the sentence exactly as it is stored in character array.

Example:
Please, enter your choice > 2

HAVE A NICE DAY

3. Reverse the sentence

Description: Reverses the sentence and displays the result.

Example: (if the input was DOGS LIKE CATS)
Please, enter your choice > 3

CATS LIKE DOGS

4. Rotate left N

Description: Rotates the sentence leftwards by *N* words.
NOTE: 1≤N≤MaxNum, where MaxNum is the number of words in the sentence.

Example1: (if the input was DOGS LIKE CATS)

Please, enter your choice > 4 2

CATS DOGS LIKE

Example2:

Please, enter your choice > 4 **5**

ERROR: Incorrect command, try again.

5. Rotate right N

Description: Rotates the sentence rightwards by *N* words.
NOTE: 1≤N≤MaxNum, where MaxNum is the number of words in the sentence.

Example: (if the input was DOGS LIKE CATS)

Please, enter your choice > 5 1

CATS DOGS LIKE

6. Sort the sentence

Description: Sorts the sentence's words in an **A**scending/**D**escending lexicographic order and displays the result.
NOTE: The default order is ascending.

Example1:(default)

Please, enter your choice > 6

CATS DOGS LIKE

Example2:(**A**scending order, if the input was HAVE A NICE DAY)

Please, enter your choice > 6 A

A DAY HAVE NICE

Example3: (**D**escending order, if the input was HAVE A NICE DAY)

Please, enter your choice >6 D

NICE HAVE DAY A

7. Search the word

Description: Searches the input words in the sentence and prints its *serial number from the beginning of the sentence* . If the input word is not placed in the sentence the program prints the suitable massage.

*Example*1:

Please, enter your choice >7 CATS

The word CATS is placed in the 3 place.

Example2:

Please, enter your choice >7 MOUSE

The word MOUSE is not found.

8. Count number of words

Description: Counts and prints number of words in the sentence.

Example: (if the input was DOGS LIKE CATS)

Please, enter your choice >8

The number of words in the sentence is 3.

9. Display substring M

Description: Displays all the *M* characters long substrings of the sentence (without spaces).
NOTE: 1≤M≤MaxWordLenght, where MaxWordLenght is the maximum word's length in the sentence.

If a substring occurs more than once, then display it **only once**. Parts of substring can belong to different words.

*Example*1: (if the input was DOGS LIKE CATS)

Please, enter your choice > 9 4

Substrings of 4 length are:

CATS
ATSD
TSDO
SDOG
DOGS
OGSL
GSLI
SLIK
LIKE
IKEC
KECA
ECAT

*Example*2:

Please, enter your choice > 9 5

ERROR: Incorrect command, try again.

*Example*3:(if the input was HAVE A NICE DAY)

Please, enter your choice >9 1

Substrings of 1 length are:
A
D
Y
H
V
E
N
I
C

NOTE : character A appeals 3 times in input sentence, but it's been displayed once.
character E appeals 2 times in input sentence, but it's been displayed once also.

Program design:

You will ***have to*** implement ***at least*** the following functions:

1. `void printMenu()`

   The function printMenu prints user menu.
   The output of this function MUST match output of example program **EXACTLY.**

2. `int checkChars(char str[])`

   The function checkChars checks if input sentence contains at most 30 characters.
   If so, it returns 1, otherwise, it returns 0.

3. `int maxWordLength(char str[])`

   The function maxWordLenght computes and returns the maximum word's length in character array *str* given as parameter .

4. `int countWords(char str[])`

   The function countWord computes and returns the number of words in the sentence holded in character array *str* given as parameter.

5. `void inputSent(char str_in[],int k,char str_out[],int n)`

   The function inputSent receives input array of characters *str_in* and its size *k*, prints it *exactly as it entered* and assigns it to the array *str_out* of size *n*.

6. `void printSent(char str[],int n)`
   The function printSent prints the sentence holded in character array *str* given as parameter.

7. `void revSent(char str[])`

   The function revSent reverses the sentence holded in character array *str* given as parameter.

8. `void rolSent(char str[],int m)`

   The function rolSent receives the sentence holded in character array *str* given as parameter and rotates it **leftwards** by *m* words.

9. `void rorSent(char str[],int m)`

   The function rorSent receives the sentence holded in character array *str* given as parameter and rotates it **rightwards** by *m* words.

10. `void subSent(char str[],int k)`

The function subSent receives the sentence holded in character array *str* given as parameter and prints **all** *k* characters long substrings of the sentence(without spaces).

Important notes :

- Please notice that **every function should change the stored string**. All examples in this file, exemplify the function call immediately after the original string was received. **It is very important to store the updated string after each function call.**
- The output of your program MUST be **exactly** as in the examples here!
- Don't forget to check that the input data are valid and not out of the defined range. If input invalid then notify the user and ask to enter input again.
- Don't forget to write clear comments, a nice presentation of the program, meaningful variable names.
- Also, do not use anything you didn't learn in class. This includes pointers, global variables, dynamic memory allocation, libraries, files, etc...

Enjoy!