CMPE 326 Concepts of Programming Languages

Spring 2020

Homework 2

Due date: 27/04/2020 23:59

In this assignment, you are going to write a program in C that manages your personal notes with tags. Using the program, you can not only enter your notes, but also tag them for a better organization. You can search your notes via tag expressions, which are various tags combined with boolean operators like and, or, not.

Input/Output Specification

The user may enter following commands.

- ADD: This command is used to add a new note to the system. It is followed by a positive number selected by the user as the **id** of the note. It is the user's responsibility to enter unique ids for all notes. After entering `ADD` and id number of the note, the user starts entering the content of the note. Each note is composed of *at least one* line of text. A line of text can be *at most* 500 characters, but a note may have as many lines of text as the user wants. After entering the last line of text, the user enters `END` in a new line. Here is an example usage of ADD command for entering a note with the id 3 and the first line "Things todo for the 326 Hw !".

```
ADD 3
Things todo for the 326 Hw !
 * review dynamic memory allocation
 * define the structs
END
```

- TAG: This command is used to tag one or more notes. Tags are names that are composed of at most 100 alphanumeric characters (spaces, symbols like parenthesis, mathematical operators, or punctuation marks are **not allowed**). The **tag name** comes after `TAG` and it is followed by a list of note ids ending with -1. For instance, the below command tags notes with ids 3, 12, and 40 as 'todo' (the intension is that the user sees these notes as todo items).

```
TAG todo 3 12 40 -1
```

- DISPLAY: This command is used to display contents of a note. The user enters the id of a note after `DISPLAY` in order to see the contents of the note. Your program must first print the id of the note (in the form '`Id:  <id>`' as shown in the below example) and show all its content (without changing anything when the note first added to the system).

```
DISPLAY 3
Id: 3
Things todo for the 326 Hw !
 * review dynamic memory allocation
 * define the structs
```

- FIND: This command is used to find notes according to a *tag expression*. Tag expressions are of the form $and(t_1 \ldots t_n)$, $or(t_1 \ldots t_n)$, and $not(t_1)$. Each t_i is either a tag name or another tag expression (i.e., there can be nested tag expressions). The semantics of tag expressions are defined using the usual semantics of connectives *and*, *or*, and *not*. Check the below FIND command examples. Pay attention to the syntax of tag expressions; all tag names, closing parenthesis, `AND(`, `OR(`, and `NOT(` are separated by **one space**.

  - `FIND AND( todo )` ⇒ the system must find all notes that are tagged as 'todo'.
  - `FIND AND( todo important )` ⇒ the system must find all notes that are tagged as both 'todo' and 'important'.
  - `FIND OR( todo AND( work important ) )` ⇒ the system must find all notes that are tagged as either 'todo' or both 'work' and 'important'.
  - `FIND AND( funny NOT( work ) )` ⇒ the system must find all notes that are tagged as 'funny' but not 'work'.

  The system must display the results of a FIND command **sorted by the note id in ascending order**. It first outputs `Results:`. Then, for each note in the result of the command, there must be a line in the form '`Id: <id> <firstline>`' where `<id>` is the id number of the note and the `<firstline>` is the first line of the note's content (so, you must output only the first line not the whole content of the note). Check the below example that shows how results are displayed for a FIND command (red lines are input of the user).

```
$ ./hw2
ADD 3
Things todo for the 326 Hw !
 * review dynamic memory allocation
 * define the structs
END
TAG work 3 -1
TAG todo 3 -1
ADD 10
Don't you know about the bird?
Everybody's heard about the bird.
END
TAG fun 10 -1
ADD 20
*** Play with Lola & Luka ***
          :)
END
TAG todo 20 -1
TAG urgent 3 -1
FIND AND( work )
Results:
Id: 3 Things todo for the 326 Hw !
DISPLAY 3
Id: 3
```

```
Things todo for the 326 Hw !
 * review dynamic memory allocation
 * define the structs
FIND AND( NOT( work ) fun )
Results:
Id: 10 Don't you know about the bird?
ADD 40
Problems with pointers
----------------------
1. dangling pointers
2. memory leaks
END
TAG pointer 40 -1
TAG 426 40 -1
TAG work 40 -1
TAG new 20 40 -1
FIND NOT( OR( todo urgent ) )
Results:
Id: 10 Don't you know about the bird?
Id: 40 Problems with pointers
ADD 50
Flight details: ESB-TXL
END
TAG flight 50 -1
TAG work 50 -1
ADD 60
Vacation: Azores
* Flight: ESB-LIS
END
TAG flight 60 -1
FIND AND( flight NOT( work ) )
Results:
Id: 60 Vacation: Azores
<EOF>
$
```

There will be no syntactic or semantic errors in the input. So, you do not need to check for errors. In the course Moodle page, you can find the above example input file and its expected output. Verify that your program's output is the same as the expected output.

```
$ ./hw2 < test.in > my.out
$ diff my.out test.out
$
```

Note that having a correct output for the this example does not necessarily mean that you have a correct program. Please check your program with other example cases.

You must follow the output specification strictly. Otherwise, you will loose points.

Submission

You need to submit your C file `hw2.c` using the course Moodle page. Your code will be compiled using the `gcc -std=c99 -o hw2 hw2.c` command.

Late submissions more than 2 days will not be graded. Each late day imposes 20% penalty of the total homework grade.