Assignment 1

Data Structures and Algorithms
Pointers, Linked Lists & ADTs
Due Date : 25 January, 2024
Total Marks: 100

1 General Instructions

- The assignment must be implemented in C.
- The submission format is given at the end of the assignment.
- Grading of the assignment would not only be based on the working of the functionalities required to be implemented, but also the quality of code that is written and performance in the viva during the evaluations.
- The deadline is strict and will not be extended.
- Resorting to any sort of plagiarism (as mentioned in the policy shared) would be heavily penalised.
- Some sections of the assignment have been marked as **Bonus**. Any marks lost in the non-bonus section, can be made up for by doing the Bonus section. It must be noted, that Bonus marks would not provide extra marks in the assignment, they can just cover up for lost marks in the non-bonus section.

2 Social Media Platform ADT

The task involves defining and implementing a social media platform ADT, that can handle features related to posts, comments and replies via a set of commands through the terminal.

2.1 File Structure

The code must be made modular, consisting of the following files to define and implement the required ADTs:

1. *post.h* & *post.c*
2. *comment.h* & *comment.c*
3. *reply.h* & *reply.c*
4. *platform.h* & *platform.c*
5. *main.c*

It is allowed to make any other helper files, if necessary.

2.2 Data Types [15]

2.2.1 Post [5]

The posts on the social media platform must store the following information:

1. **Username** : Username of the user posting (Always a single string)
2. **Caption** : Caption to the post by the user
3. **Comments** : List of comments on the post, by different users

2.2.2 Comment [5]

The comments on the posts must store the following information:

1. **Username** : Username of the user commenting
2. **Content** : Content of the comment
3. **Replies** : List of replies to the comment by different users

2.2.3 Platform [5]

The social media platform ADT, must store the following information:

1. **Posts** : Lists of posts stored in some order of the time they were posted at
2. **lastViewedPost** : The last viewed post on the terminal. By view, it means the one of which the details were shown latestly. If this does not hold for any post, it is the most recently added post till now.

Note: Make Platform a global instance as it will only be created once.

2.2.4 Reply [Bonus][5]

The replies to the comments on a post must store the following information:

1. **Username** : Username of the user replying
2. **Content** : Content of the reply

2.3 Functions [60]

For each of the ADTs, implement the following functions:

2.3.1 Post [5]

Function Name	**Parameters**	**Return Value**	**Description**
createPost	*char* username, char* caption*	*Post**	Creates a post using the given parameters, returning a pointer to it

2.3.2 Comment [5]

Function Name	**Parameters**	**Return Value**	**Description**
createComment	*char* username, char* content*	*Comment *comment*	Creates a comment using the given parameters, returning a pointer to it

2.3.3 Reply [Bonus][5]

Function Name	**Parameters**	**Return Value**	**Description**
createReply	*char* username, char* content*	*Reply* reply*	Creates a reply using the given parameters, returning a pointe to it

2.3.4 Platform [50]

Function Name [Marks]	**Parameters**	**Return Value**	**Description**
createPlatform[3]	—	Platform* platform	Create an instance of the platform data type. Only one such instance should be made through out the code
addPost[5]	*char* username, char* caption*	*bool posted*	Create a post of the given parameters (by calling the previous implemented function) and adds it to the existing list of posts, returning whether the process is successful or not
deletePost[7]	*int n*	*bool deleted*	Deletes the n^{th} recent post, returning whether the deletion is successful or not. Also, it should clear the comments and replies as well to the post
viewPost[5]	*int n*	*Post * post*	Returns the n^{th} recent post, if existing. If it does not exist, a *NULL* pointer must be returned
currPost[3]		*Post * post*	Returns the *lastViewedPost*. As described earlier, this post will be the most recent post, if no other post has been viewed. If no post has been done, a *NULL* pointer must be returned

Function Name	**Parameters**	**Return Value**	**Description**
nextPost[5]		*Post * post*	Returns post which was posted just before posting the *lastViewedPost*. If the *lastViewedPost* is the first post to be added, then return it. In case of any error, a *NULL* pointer must be returned. Doing this operation, will change the *lastViewedPost*, by it's definition
previousPost[5]		*Post* post*	Returns post which was posted just after posting the *lastViewedPost*. If the *lastViewedPost* is the most recent to be added, then return it. In case of any error, a *NULL* pointer must be returned. Doing this operation, will change the *lastViewedPost*, by it's definition
addComment[5]	*char* username, char* content*	*bool commented*	Adds a comment to the *lastViewedPost*, returning whether the addition is successful or not
deleteComment [7]	*int n*	*bool deleted*	Deletes the n^{th} recent comment to the *lastViewedPost*, returning whether the deletion is successful or not. Also, it should clear the replies to the comment as well
viewComments [5]		*Comment* comments*	Returns a list of all comments to the *lastViewedPost*. The order of the comments in the list must be in order of the time of commenting, the latest being at last. The order of replies should be the same as well
addReply [Bonus][5]	*char* username, char* content, int n*	*bool replied*	Adds a reply to the n^{th} recent comment of the *lastViewedPost*, returning whether the addition is successful or not
deleteReply [Bonus][5]	*int n, int m*	*bool deleted*	Deletes the m^{th} recent reply to the n^{th} recent comment of the *lastViewedPost*, returning whether the deletion is successful or not

3 Sample I/O

Input	**Output**
create_platform	
add_post user1 caption1 add_post user2 caption2 add_post user3 caption3 delete_post 2	
view_post 2	user1 caption1
add_post user4 caption4	
current_post	user1 caption1
previous_post	user3 caption3
previous_post	user4 caption4
next_post	user3 caption3
add_comment user1 content1 add_comment user2 content2	
view_all_comments	user1 content1 user2 content2
delete_comment 2	
view_comments	user2 content2
current_post	user3 caption3
add_comment user3 content3 add_reply user1 reply1 2 add_reply user2 reply2 2	

view_comments	user2 content2
	  user1 reply1
	  user2 reply2
	user3 content3

delete_reply 2 1	
view_comments	user2 content2 user1 reply1 user3 content3

Note: Please note that all inputs, even for a single command would be given in separate lines. This is being done to make it easier to take input easily.

3.1 Driver Code

Write a C program which illustrates the above operations by building a good command line interface, i.e. ask the user which operations he/she would like to do for the user. This would be the *main.c* file.

3.2 Note

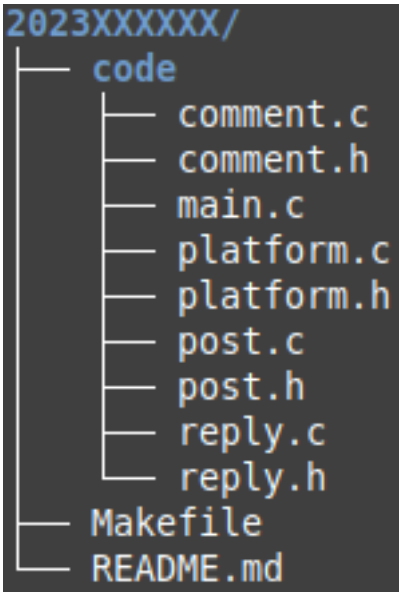
- The reply functionalities in the above Sample I/O are only for the people who have attempted bonus.
- If the required post, comment, or reply does not exist, output a valid error-handling message on the terminal.
  Example: If the nextPost does not exist, output "Next post does not exist."

4 Evaluation

- There will be manual evaluation for this problem. However, your function signatures, return types and return values should match exactly as the ones described above.
- 20 marks are for Viva which will involve explaining your code properly.
- 5 marks are for maintaining code modularity and following good coding practices.
- The bonus is of 20 marks in total. It will **NOT** fetch you any additional marks.However doing the bonus will help you compensate for the marks you lost in assignment.

5 Submission Format [5]

Submit a zip file, *roll_number.zip* which has a folder named as your roll number. The contents of the folder should be as shown below assuming 2023XXXXXX as a roll number:



The *Makefile* must compile the code files inside and a *README.md* should be made containing assumptions taken, if any. Any other helper files maybe included as well.

Follow good coding practices i.e. make your code modular. Implementing all the functions in a single file will lead to penalties.