HW4: Linked List-Self Referential Structure

A linked list can be defined as a self-referential structure that has a pointer to a structure as a member element as shown below:

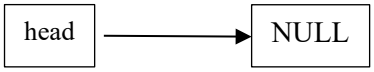
```
struct node{
    char data[40];
    struct node* next;
};
```

Here, the struct template **node** is a linked list which has member variables **data**, an array of characters and **next,** a pointer to a variable of type node.

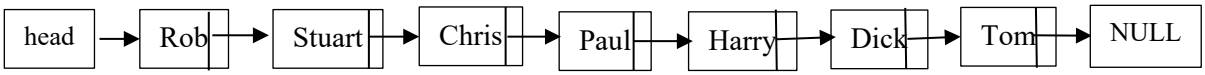
In the attached test file named "**LinkedListString_test.c**", an empty node "head' is created.

struct node* head = NULL;

At the beginning of insertion of any node, the list looked like follows:



After insertion of seven nodes, the linked list now looks like the following:



You need to implement the following functions:

BASIC PART: **20 pts**

/* Function to count the number of elements in the linked list */

int listCount(struct node *head){ int count = 0; return count;} **3 pts**

/* Function to reverse the linked list */

void reverseList(struct node head_ref){}** **4 pts**

/*Function to delete a particular element/ multiple occurrences of an element in a linked list*/

void deleteElement(struct node **currP, char *value){} **6 pts**

/*Function to delete all elements in a linked List */

void listAllDelete(struct node **currP){} **3 pts**

As you can see, **listCount** will count the number of nodes in the linked list, **reverseList** will reverse the nodes in the linked list, **deleteElement** will delete a particular node with the given value for data, and lastly, **listAllDelete** will delete all the nodes in a lineked list.

After you complete writing the functions as mentioned above and uncomment the different function calls in main, when you run the program, it gives the following output:

```
syasmin@cscd-linux01:~/CSCD204/HW4$ ./LinkedListString
Linked list after insertion:
Harry  Harry  Rob  Stuart  Chris  Paul  Harry  Dick  Tom

Number of elements in the list: 9

Linked list after deletion of a node with value "Harry":
Rob  Stuart  Chris  Paul  Dick  Tom

Number of elements in the list after deletion: 6

Reversed Linked list
Tom  Dick  Paul  Chris  Stuart  Rob

Number of elements in the list after deletion: 0
```

Makefile: **4 pts**

You need to use **Makefile**; split the attached file into three files:
(a) a header file named **LinkedListString.h** and
(b) two separate source files: **LinkedListString .c** and (c) **main.c**.

- **LinkeListString.h** will only contain template and function declarations. For convenience, you can use additional functions.
- **LinkeListString.c** will contain function body.
- **main.c** will include the main function only.

BONUS PART: **5 pts**

5 extra bonus points will be given to those who will implement the following extra functions.
This function will insert a node in a sorted linked list.

void insertElement(struct node **currP, char*value);

So, at first, you need to sort the existing list on the basis of the data in each node.

void sortLinkedList(struct node currP);**

Inside your sort function, you need to implement swap function.

void swapData(struct node* nOne, struct node* nTwo);

Insert operation will insert node according to its value. Here's some output:

```
syasmin@cscd-linux01:~/CSCD204/HW4$ ./LinkedListString
Linked list after insertion:
Harry  Harry  Rob  Stuart  Chris  Paul  Harry  Dick  Tom

Number of elements in the list: 9

Linked list after deletion of a node with value "Harry":
Rob  Stuart  Chris  Paul  Dick  Tom

Number of elements in the list after deletion: 6

Reversed Linked list
Tom  Dick  Paul  Chris  Stuart  Rob

Inserting an element with data "Ann" in a sorted list:
Ann  Chris  Dick  Paul  Rob  Stuart  Tom

Inserting an element with data "Ann" in a sorted list:
Ann  Ann  Chris  Dick  Paul  Rob  Stuart  Tom

Inserting an element with data "Zoe" in a sorted list:
Ann  Ann  Chris  Dick  Paul  Rob  Stuart  Tom  Zoe

Number of elements in the list after deletion: 0
```

Submission:

A zip file containing two folders: Basic and Bonus (optional):
Basic part includes:

- LinkedListString.h
- LinkeListString.c…….
- main.c and
- Makefile
- Submit the output capture as a file named LinkedListStringHW4outBasic.pdf – containing at least 3 different runs.

Bonus part (optional) includes:

- LinkedListStringBonus.h
- LinkeListStringBonus.c…….
- mainBonus.c and
- Makefile
- Submit the output capture as a file named LinkedListStringHW4outBonus.pdf – containing at least 3 different runs.

Name your zip file with your last name first letter of your first name HW4.zip (ex: yasminsHW4.zip)

Submission deadline is: 11:59 pm, Tuesday, November 24. No late submission will be considered.