CIS*2500: Lab Assignment 4

Linked Lists

Using the following struct (where 'NODE' can be called anything you want, except for 'Node')

```
typedef struct NODE {
    int value;
    double key1;
    double key2;
    struct NODE * next;
    struct NODE * sort1;
    struct NODE * sort2;
} Node;
```

Write a function that takes in an integer and produces a "random" linked list
where for each node:

- produce a random int between 0 and 10 stored in `value`,
- a random `double` between `10.0` and `50.0` stored in `key1`
- a second random double between `50.0` and `90.0` stored in `key2`
- create a linked list, using the `next` pointer to link the list.
- Set the other two `Node *` links (`sort1` and `sort2`) to `NULL`.

Have a pointer called `head` pointing to the first node in the above linked list

Create a function that traverses the above linked list, following the `node->next` links,
and prints out a node per line using the following format:
`< node->value, node->key1, node->key2 >`

Create a function that uses the `sort1` Node pointer to link the nodes in ascending order according to `key1`. To do this create a node pointer called `sort1_head` initially set to `NULL`.

Traverse the linked list using `node->next` and insert in sort order using the `node->sort1` link as well as the `sort1_head`, which should be modified if the node is the smallest found so far.

Note: `node->next` links should be left unchanged;
i.e. if followed, the original list order will be traversed.

Create a function that traverses the above linked list, following the `node->sort1` links,
and prints out a node per line using the following format:
`[ node->value, node->key1, node->key2 ]`

Create a function that uses the `sort2` Node pointer to link the nodes in ascending order according to `key2`.
To do this create a node pointer called `sort2_head` initially set to NULL.

Traverse the linked list using `node->next` and insert in sort order using the `node->sort2` link as well as the `sort2_head`, the latter of which should be modified if the node is the smallest so far.

Note: `node->next` and `node->sort1` links should be left unchanged;
i.e. if followed, the original list order, or sort order by `key1`, will be traversed.

Create a function that traverses the above linked list, following the `node->sort2` links,
and prints out a node per line using the following format:
`{ node->value, node->key1, node->key2 }`

To test the above functions, write a main that

- Ask the user for an `int` greater than 5 to determine the number of nodes to be created.
- Create a 'random' linked list as specified above linked to using a `Node` pointer called head.
- Create a node pointer called `sort1_head` that holds the head node of the above linked list when following the `node->sort1` links, which should be in ascending order according to `key1` values.
- Create a node pointer called `sort2_head` that holds the head node of the above linked list when following the `node->sort2` links, which should be in ascending order according to `key2` values.
- Print out the linked list using
    - the `next` links
    - then the `sort1` links
    - then the `sort2` links
- Traverse the next links and for every third node replace the nodes value by its current value times 10 (do not change key1 or key2) .
- Again, print out the linked list using
    - the `next` links
    - then the `sort1` links
    - then the `sort2` links

NOTES (IMPORTANT INFO ... READ CAREFULLY)

1. Your code must compile cleanly with no error or warning messages using the -Wall flags in gcc.
2. Your source code should be properly formatted and meaningful variable names should be used.
3. Your source code should contain brief comments describing the functionality and the major components of each procedure. Any complex structures should also be commented.
4. If you hand in an assignment that does not compile you will get a mark of zero.
5. All work in this course is to be done independently.
6. This Lab will be graded in person during your scheduled lab grading period. If you are not graded in person by one of your assigned TAs (or TA scheduled along side them), you will receive a 0.
7. You should also submit your source code file (.c and .h files) and a makefile to the L3 submission link in the course website. Note: this is not a substitute for in person grading.