2413 Assignment 5

Due Date
2-29-16 11:59pm

For this assignment you will be implementing both a merge sort and a quick sort on doubly linked lists. I recommend that you define the doubly linked list structures and associated functions in its own header so that main and the implementation of those functions can remain separate. Create any and all auxiliary/helper functions you need or want.

Input Pattern:

```
Num_0 Num_1 Num_2 Num_3 ... Num_(Num_0) SortRoutine
```

Sample Input:

```
5 10 8 2 4 6 mergesort
```

Sample Input:

```
10 10 8 2 4 6 5 3 1 9 7 quicksort
```

The first integer is how many integers to read in the sequence.

Sample Input:

```
5 10 8 2 4 6 mergesort
10 10 8 2 4 6 5 3 1 9 7 quicksort
```

Your input should handle reading in a sequence of sort problems until end of file is reached.

Main Loop PseudoCode

1. While able to input a number D
   (a) Read D numbers and put each number at the back of a doubly linked list .

(b) Read the name of the sort routine from input.

(c) Print the numbers to output in order on the linked list (same order as input).

(d) perform the sort indicated in the sort routine.

(e) Print the numbers to output in order on the linked list (should now be sorted).

(f) Free the nodes on the linked list.

1 Merge Sort and Output

Merge Sort Pseduocode

1. Get the size of the list, and `printf("Merge Sort, Size Of List: %d", size)`
2. If the input list is empty or has a single element, return
3. Split the input list into two new lists, A and B.
   (a) In order for your output to match the expected output, use the following algorithm for splitting.
   (b) set a temp pointer to the head of the list.
   (c) advance the temp pointer (size-1)/2 positions
   (d) temp points to the end of A and temp-¿next points to the start of B
   (e) Finish splitting the original list into A and B so that they are well-formed but separate lists.
4. MergeSort(A)
5. MergeSort(B)
6. Merge A and B (now sorted) together.
   (a) While A and B are non-empty, remove the smaller head node and put it at the back of the input list.
   (b) If either A or B are non-empty, append the remaining nodes to the end of the input list.
7. sanitize head and back pointers of A and B and free them.
8. `printf("DONE: Merge Sort, Size Of List: %d ", size)`
9. return

2 Quick Sort and Output

In order to get consistent behavior on output, we will not be choosing a random element to be the radix or pivot.

Quick Sort Pseduocode

1. Get the size of the list, and `printf("Quick Sort, Size Of List: %d", size)`
2. If the input list is empty or has a single element, return
3. remove the head node on the input list as the radix/pivot.
4. `printf("Picked Pivot: %d", pivot)`
5. while input list not empty, remove head node and append to back of A or B as follows:
6. List A contains all other nodes that are $\leq$ the radix.
7. List B contains all nodes that are $>$ the radix.
8. Quicksort(A)
9. Quicksort(B)
10. Concatenate [A, radix node, B] together in the input list
11. sanitize and free the two new lists created now that they are empty and we are done with them.
12. `printf("DONE: Quick Sort, Size Of List: %d ", size)`
13. return

3 Examples

Example 1 Input To Program:

```
5 10 8 2 4 6 mergesort
```

Example 1 Output:

```
10 8 2 4 6
Merge Sort, Size Of List: 5
Merge Sort, Size Of List: 3
Merge Sort, Size Of List: 2
Merge Sort, Size Of List: 1
Merge Sort, Size Of List: 1
DONE: Merge Sort, Size Of List: 2
Merge Sort, Size Of List: 1
DONE: Merge Sort, Size Of List: 3
Merge Sort, Size Of List: 2
Merge Sort, Size Of List: 1
Merge Sort, Size Of List: 1
DONE: Merge Sort, Size Of List: 2
DONE: Merge Sort, Size Of List: 5
2 4 6 8 10
```

Example 2 Input To Program:

```
5 10 8 2 4 6 quicksort
```

Example 2 Output:

```
10 8 2 4 6
Quick Sort, Size Of List: 5
Picked Pivot: 10
Quick Sort, Size Of List: 4
Picked Pivot: 8
Quick Sort, Size Of List: 3
Picked Pivot: 2
Quick Sort, Size Of List: 0
Quick Sort, Size Of List: 2
Picked Pivot: 4
Quick Sort, Size Of List: 0
Quick Sort, Size Of List: 1
DONE: Quick Sort, Size Of List: 2
DONE: Quick Sort, Size Of List: 3
Quick Sort, Size Of List: 0
DONE: Quick Sort, Size Of List: 4
Quick Sort, Size Of List: 0
DONE: Quick Sort, Size Of List: 5
2 4 6 8 10
```