CIS*2500 W20 Assignment 4

Questions & Answers Part 4

“I am working my way through A4Q1b, and have noticed you do not specify what the mapping/reducing function pointer arguments should be. I understand that this is intentional, however this results in two functioning solutions: take in the whole Node structure as an argument or take in only the value_t field as the argument. I looked to the notes for clarification, but here it is also ambiguous. On the last two slides, you define reduce() as the following:

```
value_t reduce(List * list, value_t (*reduce)(value_t, value_t), value_t init )
```

This suggests that you want the function pointers to accept value_t arguments. On the final slide, however, reduce is called with a function that that takes (Node *, Element) arguments and again with a function that takes (Node *, size_t) arguments.

Could you please clarify what arguments these mapping and reducing function pointers are intended to use?”

The lecture notes had a "typo" (actually old code that I forgot to change).
I have corrected it now.

To summarize the changes, both

`fn(Node *, Element)` and `fn(Node *, size_t)`

should have been

`fn(value_t, value_t)`

or, now for the assignment,

`fn(value_type, value_type)`

Mark Wineberg

“I read your latest Q&A post and I have a question about one of your responses. You said that for Q1 "When you free a node that has value_type as a char *, then you have to free the string memory. This is not necessary when value_type is an int". However, my current implementation works for both ints and char arrays with no special freeing of string memory or any leaks. The only mallocing I do in Q1a is for the creation of Nodes and the List -- I do not malloc strings separately.”

Now that you point it out, I remember that I designed against the need to free the string inside a node.

This is accomplished by demanding an 80 char limit on the strings in the Node struct
so it doesn’t have to be malloc'd.

I will correct my advice.

Mark Wineberg

“Do we need “<“ in command line before the file name or before the filename while entering input when there is no filename in the command line argument?”

Neither and both, simultaneiously. Here is how to think about IO redirect:

Linux> a4q1a_char LewisCarroll.input
Program should open the file LewisCarroll.input to use as the file pointer.
Input is from the file
Output is to the terminal aka the console;
    i.e. the "screen" in the window that accepts Linux commands

Linux> a4q1a_char
Program should use stdin as the file pointer.
Input is from the keyboard. Input is terminated with ^d (which translated to EOF)
Output is to the terminal.

Linux> a4q1a_char < LewisCarroll.input
Program should use stdin as the file pointer.
Input is from stdin which is fed from the file LewisCarroll.input
Output is to the terminal.

Linux> a4q1a_char < LewisCarroll.input > q1a_LewisCarroll.output
Program should use stdin as the file pointer.
Input is from stdin which is fed from the file LewisCarroll.input
Output is to the file q1a_LewisCarroll.output. You can name the file anything you want.
You do not have to change your program to do this. You are using stdout to print to.
    This just redirects it from the screen to a file
The order of "< filename" and " > filename" does not matter.

Linux> a4q1a_char LewisCarroll.input > q1a_LewisCarroll.output
Same as the first command, except the output is sent to a file instead of the screen.

Note: The number of spaces does not matter.
I just added more to separate the terms for easy reading.

Mark Wineberg

PS I have not included, in the above discussion, the "| next_program" construct, called a 'pipe'. Here stdout is sent to the stdin of another program, which is run next. You can chain as many of these as you want, with the final one outputting to the screen or to a file with '>'.

Sir do we need to write linux> too???

No. This represents the prompt.

Each machine is set up with a different prompt. I do not know what yours looks like.

The third command list looks like the following on my computer

```
markwineberg@Marks-MacBook-Pro ~ %  a4q1a_char  <  LewisCarroll.input
```

Mark Wineberg

"I am having a hard time implementing filter recursively with only being able to pass one list to the function filter. Wouldn't I need access to both lists? Not just 1 list and a function pointer?

Example,
```
     Sorted_list* filter (Sorted_list* list, Sorted_list* new_list, (*fn_ptr)(Node*))
```
Instead of
```
     Sorted_list* filter (Sorted_list* list, (*fn_ptr)(Node*)
```
Let me know."

Yes, you will need a function that has both lists as arguments, one to read from, one to add to. However, it should not be the filter function itself, otherwise you are exposing the inner working of the function to the user.

This means you need a "helper" function. So filter(list, filter_fn) calls another function filter_helper(Node * source, Node * target, filter_fn), which can then be recursed in the way you suggest.

Mark Wineberg

“Might I lose marks on A4 for assuming a maximum length for user input commands?”

No. In fact, there is naturally a maximum length based on the way the questions are formed.

Just state your assumptions a reasoning in your read.me

Mark Wineberg

“In question 1a, can we assume that the .input file will be correctly formatted, and that each line will have < 80 characters? Do we have to make our input system (ie putting the lines into the linked list) able to handle irregular input, punctuation, etc.?”

No, keep it simple.

Just put all assumptions in the read.me

Mark Wineberg

“I am wondering if the input commands in the file given are accepted and tested then inputted into the list? For example, if the command "p i like dogs" in the file, then do we compare the command with every possible command that could be used that was outlined in the instructions to what matches then execute the appropriate function?”

Yes.

This can be done using "cascading ifs". There aren't that many commands.

Mark Wineberg

“For the shorthand command for the function "destroy_list" when getting it from the input file, is it "destroy". For example in the file there would be a line that says, "destroy"

No, there is no command to destroy a list.

You would just use it at the end of your main.

Mark Wineberg

“In the Q and A, you repeatedly assert that several function (such as size and the remove functions) cannot fail, however, what are we supposed to return if the Sorted_List pointer itself is NULL? (This would mean that there is no head or tail to access)?”

You are right about remove. I will correct my Q&A

For size, if head == NULL, size == 0

Mark Wineberg

On rereading your question, I think you were only questioning the remove function, not size.

So you are just right :-)

Mark Wineberg

“I was wondering about the push function. In the lecture notes for the doubly linked list push function is_empty was used in it. Do we have to use this as well, even though it is not included in the instructions? Thank you.”

You could ...
or you could use size(list) == 0 if you don't want to write the extra function.

Mark Wineberg