CIS*2500 W20 Assignment 4

Questions & Answers Part 5

“My computing cluster and I are unsure what “If any ADT implementation details are exposed and used directly in a command implementation, you will not receive any marks for that command” means.

I finished the assignment, but I am very worried that I might be “exposing ADT implementation details”

Can you please explain this more?”

An "Abstract Data Type" is defined by its data type name and functions that act on the data of that type. The details of the abstract data type should be completely hidden from the user (who is the programming using the data type and functions).

For example, we should be able to change the Sorted_List ADT so that the struct stores an array instead of a linked list. The functions would then deal with array manipulation rather than linked list manipulation.

Outside the inner working of our ADT, however, nothing would change. The functions would behave the same way. The sorting would be done on push or append. We can use map and reduce using exactly the same code.

This idea is captured by the phrase "the usage of the ADT is implementation invariant". So we could completely change the way the ADT is processed without changing a single line of code that has used the ADT.

So we cannot return a Node or Node pointer, or require them as arguments (for public functions) since that has to do with the Sorted_List implementation. These nodes make no sense if we implemented the ADT as an array, which has no nodes.

We also cannot use any "internal" functions from the implementation of the ADT.

So when writing the commands, you have to use the functions that take a Sorted_List argument and or value_type and key_type arguments. You cannot use nor return any Node values or next pointers or sort pointers or anything of that sort.

Hope this clarifies everything.

Mark Wineberg

“Can I still use a Node pointer when setting my function pointer?

For example,

```
if(rem_mixed) {
     int (*fract_ptr)(Node*);
     fract_ptr = ignore_mixed;
     Sorted_list* new_list = filter(list,fract_ptr);
     print_all(new_list,mode);
     destroy_temp_list(new_list);
}”
```

No!!!

The function has to take arguments of type "value_type".

You should not be using Node * node (where I assume you are going to access node->value), anywhere outside of internal ADT code.

This is a perfect example of the type of mistake I was warning you against.

If Sorted_List were implemented using

```
typedef struct {
   value_type * data;
   int * sort_index;
   int head;
}
```

Then int (*fract_ptr)(Node*); makes no sense. I should be able to change the implementation of the Sorted_List ADT without having to change anything in the code you write to processes fractions.input.

Great example.

Mark Wineberg

“I was just wondering if we are supposed to account for blank lines being added to the list. For example, is this going to be a valid test?

```
a this is my test file
p
a this is still my test file”
```

Yes, you need to consider this.
The LewisCarroll.input test file has such a line in it.

Mark Wineberg

“For the most part my q1a is working, but for the char version, strange things are happening, that value of the string before passing into the append/push is as it is in the file. however, as soon as it is passed into the function, if I print what i being passed in, it is a bunch of junk/numbers. I don't know why this is happening or how to fix it.”

There could be a many possible problems, but without seeing your code, only two occur to me, both of which don't quite fit with your explanation:

1) Not using `strcpy`

Perhaps, you are just copying the string pointer without doing strcpy to copy the contents of the string. However, this would set everything to the input string you are using, not give you garbage.

If you do have an array of characters in place for value (see my second possibility) then you might be copying the string pointer into the first character, or maybe four characters. However, this would cause a type mismatch and give you an error unless you did a `(char)` cast on the string pointer.

If you did do the cast, you would get garbage, but requires extra coding (casting to char) that wouldn't be there for the other typedef's for `value_type`. Are you keeping your implementations of linked list code separate and different for the various different questions/programs? Otherwise why would you be casting to char?

2) incorrect setup of the `typedef`

if `value_type` is an `int` the `typedef` looks like this:

```
typedef int value_type;
```

For `char [80]` however, it looks like this

```
typedef char value_type[80];    /* putting the [80] before value_type wouldn't compile */
```

If you used `typedef char * value_type;` this would not create the 80 places in the Node `value` field. You would then have to malloc it separately (and free it as well), which you may not have done.

If you didn't malloc, it will hold a random memory pointer. If you strcpy the characters there, you could get a seg fault or if it happens to, by coincidence, hold a pointer you have access to, and strcpy there, it might get overwritten by who-knows-what later on. This might give you the garbage you are seeing, but I would think a seg fault would be more likely.

Do any of these two possibilities look like something you did?

Mark Wineberg

Note: the problem was not using strcpy

"I was working on the main for a4q1a_char and noticed that just by doing sscanf to the input only grabs the first word of the sentence. How can you grab the rest of the words in the sentence and store it into one variable? Would there need to be another loop of some sort or is there a trick that would make it shorter?"

You use fgets() to get the entire string, then look at the first few characters to see what the command is. The rest of the string, minus the white space up to the first letter of the string, would be the value to store in the linked list.

For the programs that use an double and int parameter input (e.g. a 3.2 7), use fgets to get the string, then sscanf on the string to do break out the parameters. In these cases, you will not have the "first word only" problem.

Mark Wineberg

"Q1b: Are the map_reduce and map_2_reduce functions supposed to modify the original list(s)?"

~~Yes.~~ Definitely No!

(*I copy and pasted the wrong one-word answer while tired last night … sigh*)

"Q1b: Are the map_2_array and map_2_reduce functions supposed to work with two lists of varying length or must they have the same length?"

They must have the same length.

Mark Wineberg

"Should set_fractions be changed to

```
Int set_fraction(Fraction* fract, long num, long denom)
```

Instead of

```
Int set_fraction(Fraction* fract, int num, int denom)
```

I tried `functions.input` with `num` and `denom` as integers but my function `set_fraction` would cut off the fraction `43643064465/39093069015` until I changed it to accept `long` and then it worked."

In retrospect, your approach is a better design, not only because having the parameters be int's wouldn't work in the .input example I gave, but because it simplifies the code.

I had made the num and denom ints because I wanted the internal representation to be larger for the purpose of addition. However, with overflow checking, having the parameters be longs will save on doing internal casts to long i.e. `(long) num`.

I will change the assignment, so it be done either way. When testing, we will only use numbers of size `int`.

Mark Wineberg

“I have a question pertaining to the Sorted List abstract data type. I am currently using one structure titled "typedef struct NODE" with the datatype as Sorted_List. My question is as follows: is what I currently have correct or are we supposed to have two separate structures being the one you provided us with at the beginning of the Q1a instructions plus a separate Sorted_List struct.”

Two structures.

First for Node, second one for Sorted_List.

Mark Wineberg

“-DFRACT and -DCHAR don’t work for me so I had to make different sorted_list functions? Would i lose Mark's for using it?”

They do work. To see how, look to the template that was just posted.

Yes, you would lose marks.

Mark Wineberg

“I saw the changes they are nice and all but if I have a way that works. Do I need to change I change my program to have a commands.c and a commands.h file and a sorted_list.c? I am doing it the way that I normally run programs and create functions and it works for me, I've never seen this way before. Am I able to keep what I have and not get marks deducted off?”

The opposite.

If you do it yourself without using the template, you can state that is what you have done for a bonus +5 marks, provided it works.

Mark Wineberg

PS Nathan’s template is a nice way of structuring code. It is the way I structure my own code.

“./ program < filename should use Stdin as a file pointer , where Input is from stdin which is fed from filename But that's confusing me , I am not sure I understand what that means? Are we taking from stdin? Or a from a file? The redirect signs make no sense to me, can you please help me?”

The IO redirect is done on the Linux side.

stdin and stdout do *not* connect directly to the keyboard and terminal in C. It is Linux that does that.

C just knows that there are characters coming into stdin and going out to stdout. Where the characters are coming from C doesn't know.

When you run the program with no < , Linux takes the characters from the keyboard and sends them to stdin. If you run the program with "< filename" then Linux takes the characters from the file associated with filename (not from the keyboard) and sends them to stdin.

Similarly with no > Linux takes the characters from stdout and sends them to the screen. If you run the program with "> filename", then Linux takes the characters from stdout and stores them in the file associated with filename (not sending them to the screen).

If prog1 | prog2 is used, Linux first runs prog1, then takes the characters from stdout from prog1 and sends it to stdin of progr2, which it runs after prog1 ends.

Hope this helps.

Mark Wineberg

“But how do we code this? I really don’t know how to go about this.”

You don't.

You just have to have used stdin and stdout, which you have already done.

Linux does the connecting without your programming intervention.

If you want to use IO redirect you just run the program using the IO redirect commands. You have already done everything you need by using stdin and stdout.

Furthermore, if you look at argc when you use "prog < infilename" or "prog > outfilename" or both you will see that argc == 1, i.e. no command line arguments were passed into your program.

Again, IO redirect is something Linux does, not something you do (except by using stdin and stdout).

Mark Wineberg

“Any return of value_type (in map_fn or reduce_fn or reduce itself) seems impossible to me when it is equal to char[80], namely in the signature.
I solved this already in the following way: passing value_type* as a pointer in order to update or "return" everywhere, so that

List *map(List *, void (*map_fn)(value_type*));”
The problem you are having is that you haven't defined value_t properly for char [80].

The code should be:

```
typedef char value_t[80];
```

If you use that, you will not have to change the signatures of map, reduce, etc.

Mark Wineberg

“Just wanted to confirm with you if the remove_smallest_key and remove_largest_key functions return void, similar to the remove_first function?”

Actually, it should return and int for SUCCESS or FAILURE.

Remove can fail if there are no nodes to remove, i.e. the list is empty.

My previous Q&A answer was wrong (I was tired when I wrote it).

Mark Wineberg

“I was wondering if you will always provide a destroy or empty at the end because my program only does not contain any memory leaks if the user provides the instructions? Not sure if this was the point of them.”

Empty is a command because it is possible that the user might want to empty a list to start the list fresh for new push and append commands.

Destroy is not a command because this is something the program should do at the end, not the user in the middle ... or at all really given the concept of the program. If the user does this, they cannot use the list anymore so the program would have to end anyway. So this ability is kept from the user. When the commands end because of an EOF, the list or lists will be destroyed by you.

Really, in effect, EOF is the destroy command.

Mark Wineberg

PS Yes, they are there to prevent memory leaks. You must free all nodes and the list at the end of the program.