CIS*2500 W20 Assignment 4

Questions & Answers Part 8

"How are we supposed to obtain overflow status in question 3? I use reduce to get sum of fractions. But reduce returns fraction itself, there is no way to return FALSE which resulted from intermediate additions (moreover, the generic reduce will continue computation not looking at any errors, and they may vanish). Should we simply #ifdef FRACT and rewrite reduce version in sort_list.c"

Do not change how reduce works! While, returning "success/failure" to the function and getting the information out as a parameter to also be used inside reduce, or alternatively keep returning the result value and adding a "success" parameter, would work, this is not the standard way reduce is defined. It also would force you to add a wrapper around all functions while as it is currently defined, there are some "natural" reducing functions that could be used as is.

There is an easier solution, which you already thought, albeit in a slightly different form than the ideal solution, [redacted].

To facilitate other students in coming up with the solution, I have added the following hint to the assignment

Hint: you will have to change the Fraction struct
to indicate if you are in an overflow/underflow situation.

Mark Wineberg

"Sir when I type this command

```
Linux> ./a4q1a_int < integers.input
```

It straight away gives me the output without the "back and forth" when stdin is used without the "< integers.input". Is it fine or do I need to type the commands when using this command line argument.?

This is the way it is supposed to function.

```
./a4q1a_int integer.input
```

and

```
./a4q1a_int < integer.input
```

should behave identically the way I designed the program.

Only

```
./a4q1a_int
```

should give the "back and forth" interaction with the program.

This is how Linux works.

Mark Wineberg

“Sir in Q3 in some places where output according to your output example file is 4.0, my output is coming out to be 4.000000 is it fine or marks would be taken off?”

It is fine.

I stated in the output document for the example that your number of decimal places will not match mine, nor do they have to.

Mark Wineberg

“Does the "a " command get replaced with a|n command? Is the answer the same for all the command|n 's? If not, does the original commands just default to the first list?

Answer: handle it however you wish.

The default to a|1 works well. Another approach is to have an 11th list not stored in the array, but kept the same way you keep lists in q1a.

We actually will not be testing for that condition.

I just wanted to get the idea that the majority of the code for q1a will be applicable to q1b.

Mark Wineberg

**“For A4q2a), counting up to n recursively, should the first recursion/return be 0 or n?
Just trying to rewire my brain to use recursion instead of loops at the moment and I’m getting myself a little confused I think.”**

You need to print 0 first, then 1, then 2, etc.

Mark Wineberg

“can the input file have a "create" command? For example if u destroy the list and want to create a new one. I am asking this since i do not see a shorthand for the command”

There is neither a create command nor a destroy command.

List get created at the beginning of the program automatically and destroyed at the end, again automatically.

The user of the program is not given control over this.

The closest the get is with the empty command, but even then the list after being emptied, still exists.

Mark Wineberg

“For Q1a1, is there a reason that we need to create separate commands for the functions? For example, do we need to create a separate 'command function' called rem_last, or can we just call the original remove_last() in the main that we made earlier?”

I think you mean:

"Do we need to write separate functions for the commands?
Couldn't we just call the ADT functions directly in the main?"

The answer is both yes and no.

Yes:
You could do that, and it would work.

No:
This is bad design!

Writing everything in the main as a long main function is poor programming style.

This is the way coders coded in the 1960's and early 1970's. This changed in the 80's and 90's with the introduction of structural, OO and functional programming styles.

Unfortunately, by writing code this way, the code becomes very ungainly and hard to follow.

Debugging becomes a nightmare! You don't realize this until the programs become *very* large; at which point it starts falling apart. It becomes nearly impossible to identify and spot where the bugs are hiding. After a while, the code becomes so buggy it is unusable.

You are used to writing small programs. This will not continue. A4 is actually not a large program ... it just might feel that way, as you haven't encountered one yet in any of the first year courses you have taken.

By following the design in A4, I am transitioning you, by example, to a better coding style. A function (including main, which is actually a function), should do only one thing, and one thing well. Functions become small and special purposed ... not big sprawling things.

The purpose of the command functions is to separate the user input from the implementation of the command. The implementation of the command might be small, just a direct call to a Sorted_List function, or it might be "large", with a lot of setup and multiple Sorted_List calls.

It doesn't matter the size of the function, because it serves its single purpose - separating the input/output code from computational code for the command being implemented.

Mark Wineberg

TL;DR - please follow the assignment instructions with respect to any functions asked of you.

“we are allowed helper functions that take nodes as parameters as long as they are not called in the main right?”

Functions that take nodes as arguments are used to create the Sort_List ADT. They are used for its inner workings. For example, push(), append(), map(), reduce() etc. are all *implemented* using Nodes. These functions are the ones listed in A4 that already have Node as a parameter in their signature. They form the Sorted_List ADT.

All other parts of the program, which are not part of the Sorted_LIst ADT, **must not use** Node parameters

Mark Wineberg

“Just to be sure, I have a separate helper function that I call in rem_first and rem_last which takes a Node as a parameter and removes it from the sorted links of the linked list. Would this be allowed and count as being part of the Sorted List ADT?”

Yes, helper functions, or any function that is used to implement an ADT function, can use Nodes.

Mark Wineberg

“What do you mean by using the appropriate gcc flag? I know we should be using -Wall and C ANSI to compile on all parts, but what other flags would we need (considering the data types are constant and the formula is constant, unless the formula is so constant we should not use -Wall)?”

You are using the appropriate gcc flags.

As Fraction has been changed to `long long` from `long`, however,
you must not use -ansi as `long long` doesn't exist in the ansi version of C.

“The second question relates to how 2b will work with question 3 as the data type is long long in 3 but only long in 2b”

As you guess, the gcd function in 2b needs to be changed to `long long` in its parameters are return type. I will be posting the change shortly.

Mark Wineberg

“I cannot understand what the function pointer is being used for in the map function. I've read over the lecture notes of function pointers and the map function but I can't understand what either of them are doing from the slides alone, so any nudges in the right direction would be helpful”

The map function takes each node from a list, modifies its value and puts the result in a new node (it does not modify the original node). It does this node by node, linking the new nodes together in the same order as the nodes in the previous list.

For example, let us say we want a new list where each node is double the value from an original list,
< 4.2, 7.1, 3.4, 2.0 > would produce a new list < 8.4, 14.2, 6.8, 4.0 >

But this example is specific in the operation. What if we wanted it tripled or take the square, or take the sqrt, etc. This is where the function pointer comes in.

You pass a function in that takes a value (not a node) and returns a new transformed value. To pass in a function, you are passing a "function pointer", i.e the location in memory that is the start of the function.

So, let’s say we want to take the sqrt of the value. Now sqrt is already a function that takes a double and returns a double, so we don't have to write it.

Assuming value_type is of type double we could use map on a List list as follows

```
List sqrt_list = map(list, sqrt);
```

or we could take the sine of the values

```
List sin_list = map(list, sin);
```

We could also pass in our own created function as well.

On how to write map, you will have to go back to the slides.

Hopefully this helps.

Mark Wineberg

“When you say “You will have to change the Fraction struct to indicate if you are in an overflow/underflow situation. Do not change any of the functions in Sort_List”, are you suggesting that we add a new field to the Fraction structure definition?”

Exactly.

Mark Wineberg

"I have a few questions about the IO redirect feature we are supposed to implement in question 1a.

1. **How do we "feed" the input file into stdin --e.g. a4q1a_char < LewisCarroll.input**
2. **Do we have to print the headers and verbose commands to the output file? If we are redirecting our output to a file, and not to stdout, should we print anything at all?**
3. **How many marks are being given for the "IO redirect" formatting as opposed to just taking from a file (inc. stdin) and printing to stdout?"**

IO redirect is not a feature to implement. Linux takes care of everything as long as you use stdin and stdout.

No points as it is something Linux does, not something you program.

"So we just need to take stuff from stdin or a file and put it to stdout? Am I getting that right? Is this the same for 1b, 2, and 3?"

As long as you optionally also take the file name as a command line parameter,
./a4q1a_int integer.input
as well as
./aq1a_int < integer.input

You have it right.

Of course, it works for all questions. It is what Linux does. The only way to stop it would be to **not** use stdin and stdout, which would be going against the assignment instructions.

"What is that arrow '<' supposed to do? Does that reverse the direction of i/o?"

It is the IO redirect. It says take the input from the file instead of the keyboard when feeding information into your program through stdin

If you use "> filename" it take the information from stdout and instead of going to your screen, it goes to a file called filename (if filename exists, it is overwritten).

Mark Wineberg

"Working on A4 q3, I have stumbled upon some confusion in creating the filter function. I do not understand how it is possible to make the function recursive given the function signature provided. The reference to the list cannot be modified, nor should the function pointer. I presume that we should make the filter function invoke another function which takes a Node * argument, and this new function's behaviour would be recursive. Is this correct?"

That is correct.

Mark Wineberg

“For the sum_sq_diff function, you said "sum across all nodes into a single result". Does this mean we actually change the value inside of the node to the square difference that was calculated? Or just store it in a declared variable then add all of them up?”

Just store it in a declared variable then add all of them up.

Do not change the original list values.

Mark Wineberg

“I was looking through the slides for the reduce functions (week 9) and there is both the parent function called reduced and a function pointer passed in, which is also called reduce -- which one would take priority? Is this even allowed in the compiler? I.e is this a recursive function or just passing the values into the function that was passed in?”

This is a typo in the lecture notes.

The parameter function pointer name should be reduce_fn, just like we had map_fn and filter_fn.

I will fix it.

Mark Wineberg

PS. The code in the lecture notes actually would compile and run properly, thanks to C’s scoping rules. However, as written it could cause confusion when reading the code (a reader would think the function is recursive), so should not be written this way.

“So we call reduce_fn() in the loop, correct?”

Correct.

Mark Wineberg

“With changed types to "long long" from "long" in Q3 , the code currently cannot check for overflow, as it requires using LLONG_MAX and LLONG_MIN constants but these are unavailable with -ansi flag. It is unclear for me how to handle LLONG_MAX and LLONG_MIN with -ansi (i learnt that they are *not* equal to LONG_ on 32-bit machines, so they have be on their own), except with what I did manually: #define LLONG_MAX ~((long long)1<<63)

P.S. So the type "long long" will always be 64-bit, even 32-bit -ansi, but the problem is -- constants LLONG_ are not defined in ansi, so I have to replace them manually”

You do not need to use the -ansi flag for this assignment.

Instead, you should use the gcc flags
`-std=c99`
or
`-std=c11`

The template makefile uses -std=c11

If you do not specify a -std, and don’t use -ansi, the default is -gnu11 (virtually the same as -c11)

Mark Wineberg

PS The long long type was introduced with C99. However, it was an early change in the standard, and many ANSI compatible compilers had already introduced the feature before the C99 standard was released. Consequently, for some compilers -ansi allows long long, while others do not.

With C99, the type change was thought through, and so LLONG_MAX etc. was added into type.h, etc. This was not true with the previous, more add-hoc additions to ANSI, so very little functionality was provided to support the new type.

“First thank you for providing all the resources and Q/As but I want to hand in my assignment right now because I am at a point where I know everything works. If I end up getting further and hand in an updated version will the updated version be the one marked or how do I ensure the updated version is the one marked if I have handed in two assignments?”

The Courselink dropbox for A4 is set so when you update it with a more current submission, it will overwrite your old one. Consequently, until the deadline, you can always resubmit your assignment.

Mark Wineberg

PS Make sure that your entire assignment is in the .zip, and not just the updated files as your original files are completely overwritten and the "early part" will no long be there.

“Regarding the recursive greatest common divisor function in Q2b, would you like us to just supply the function and explanation in our readme or would you like us to include the function in our program, if so, should it be in the file with the other recursive functions used in Q2a or it's own file with a 'main' so you can test the function?”

The assignment has been changed to add a gcd command to q2.

If you implement q3 and use gcd there successfully, using gcd in q2 is now optional.

However, it shouldn't be that hard to write the gcd command for q2 if gcd itself is functioning properly. So its inclusion is recommended to make the TAs' life easier when marking.

Mark Wineberg

“Just trying to make sure I understand how we should structure our main file. Right now the way I have my code written is that in the main when the user types for example:

```
p 3.2 5
```

I call `push(list, parsed_key, parsed_value)` **function and send it the values they typed in. When they type**

```
rem_first
```

I `remove_first(list)` **function. All my ADT functions are in their own file which is included in main, but I am wondering if this is the correct idea. The admonition against use of internal code used outside of the ADT has concerned me.”**

Your description of what you have done sounds OK, provided when you say

"All my ADT functions are in their own file which is included in main"

you mean that you have included the typedefs, #defines and function signatures as a .h, and not included all of the code, by either including the ADT's .c or writing the ADT's function code and not just the signature in the .h.

You probably are not doing that as you would have been told not to in an earlier assignment, but I am just making sure.

The idea of hiding implementation details is for specifically manipulating the Node linked list directly instead of using the ADTs interface.

Of course, you have to use the Node linked list data type and functions when implementing the Sorted_List ADT. This is what is meant by "implementing" the ADT.

Anywhere else, such as when implementing an input command, you cannot directly use Node or its functions.

Mark Wineberg

“Hello, I just saw the last recommendation -- and for sum_of_sq_diff I think it is completely ok to follow old signature returning value_type, but just return -1 (this function itself is used only in a4q1b), as sum of squares of differences cannot be negative. I defined this value in a4q1b.c as #define VALUE_FAILURE (-1). Is that ok?”

Yes it is … sort of. However, you should still include the error condition inside of the Fraction struct, which is a much better general technique.

Using "impossible" values as a conduit for returning errors codes is actually a standard C trick. However, it has its limitations.

For example, if the function potentially could return any value provided by the data type, there is no value that you could use for an error code that might not be produces naturally by the function.

More modern languages use a mechanism called "exception handling" (using a technique called "catch-throw") to get around this problem. However, C predates that concept and doesn't have it.

Given the C limitations, having an error flag as part of a return data type is the best general approach, although the error code idea is often easier to implement, so is commonly used when applicable.

Mark Wineberg

“I am confused about mode in which we should print fraction when echoing (only here) a/p commands (where mode is not specified). I considered it to be SIMPLE (and 7/5 is shown as 7/5), but at the same time it must show "a 3" as "a: 3.000 3/1", not as "a: 3.000 3". So should we handle another special format (neither SIMPLE, nor MIXED, looking like SIMPLE but without printing /1 for whole numbers), or can we simply print SIMPLE? (3/1)”

The idea was to echo what was input in the command before processing by Fraction.

If "a 24/18" is entered, what is stored in the list is 4/3, but I wanted 24/18 printed for the 'a' command. So the echoing for 'a' is not the same as printing SIMPLE.

Based on this, if "a 3" is entered, the echo should be "3.0 3" as it has not yet been stored in fraction. The key is being displayed as it has to be computed at that point, and this is consistent with q1a char where the string length is computed and printed out for 'a' and 'p'

Mark Wineberg