2017/07/05 WED TLAP

- ask Jim about git.ignore for binary files (compiled C++ files). → Although, I think I like the idea of putting them all in their own folder and ignoring the contents there... but also I could ask Jim...

- So, I could go and resume work on LL (which is valuable but might not feel like forward progress), or I could move forward to ch. 5. I'm pretty tempted to move forward. I suppose I could always come back to LL later... Also, since my current objective is just to get my feet wet w/ C++, I suppose I could go way back to an easy problem just to give myself familiarity again. Not a bad idea...

- student name . . . . text
- student ID . . . . . #
- student final grade . . # range 0-100

Classes are merely extra layers of abstraction. Good old fashioned functions and variables seem to live inside classes, just w/ some different wrappers and usability details. But @ the end, they're not so different.

- records can be added or removed
- display the record of a particular student (note, author used singular)
  - identified by ID and w/ grade displayed as a number and letter (use conversion chart)
- display average score for the class

5-1 data for automobile stored in class

- manufacturer name — String
- model name — String
- model year — integer

Create class w/ get/set methods for each data member
be mindful of member naming: make sense, be consistent

===

Looking ahead to tomorrow:

- continue to figure out constructors
- try to find ways to get my basic code to work
- read/re-read TLAP 5 + online sources to better understand class
- be mindful of web app → I want a balanced approach. In a 'normal' week, consider spending 2 days on app and 2 days on TLAP? as a default? or maybe 1 on TLAP, 2 on web app, and 1 on supplemental research? I don't know what makes sense, but I'm sure I'll figure it out w/ time. It's just not 'all' or 'nothing': tabling a curiosity hole for a later time doesn't mean I've abandoned it! But since I only just got started on an actual TLAP exercise late in the afternoon today, I think it's easily justify imagining spending tomorrow on it too. ☺

2017/07/06 Thursday

sweet tunes, yo!
ok, starting up a new day w/ energizing biz! Let's do this!

- Class must always have two arguments: constructor and destructor.
  → if not actually needed, I guess the compiler can create default ones for me?
  - when programmer declares class, the constructor is automatically called.
  - destructor isn't called till program's done, class reaches end of scope, or memory deallocated.
  - neither c nor d return arguments.

- definitely ask Jim about this pointer error:
  invalid conversion from 'const char*' to 'char' → set_mfgr(" ");
  ^

- So, now what?? I seem to be sticking in one way or another in the exact same spot: the compiler doesn't like when I try to assign a value to the ☆ class variable, regardless of char or int or copied-from-internet code. What to do? I can't efficiently solve this on my own. Should I play more w/ classes (reading up on pointers, say?), or move back to my webpage where I can actually be efficient? ☆ I'm just torn enough ☺ I can't make a quick decision. I've got a little less than 45 min left in this segment of morning coding. Too much time to spin my wheels on classes, but kind of short for website. But I think swapping to the website makes sense → it's good 'transition' practice and saves me from continuing to bash my head in w/ classes. Oy! Classes!

2017/07/07 Friday

where did I leave off yesterday and where would I like to go today? I kind of ended things abruptly, so I didn't properly consider what to do looking forward... Oh yeah, Jim showed me how to more properly set up C++ file structures. That was cool.

- Ok, so it seems ☺ a good next step would be to use strings instead of chars. That maybe means I have to think about pointers too? TLAP doesn't even go there, & so I'm a little confused. But I'll plod ahead anyways to see if I can learn anything. I could also create a separate, super-simple file ☺ reintroduces me to strings w/o messing w/ this class stuff, if I think ☺ makes sense.
- my • validation step fails if the year isn't within the range: it prints random #s to screen -- presumably memory location or something. At this point it's maybe better not to have any data validation, but perhaps later I could play more w/ this idea... → hmm, maybe it's an actual value of garbage? unknown... but strange and requires more thought than I want to give just now...

→ I'd like my class variables to actually be a struct array like on p. 69 of TLAP
→ how to do ☺? Do I already have it set up??

- Ok: thoughts: find functioning 'String' to use as example in this project
  - explore 'array' functionality of struct, & class.
  - playing w/ pointers.
  - perhaps go "backwards" and work on just strings w/o classes...
  - figure out the datastructure of vars in class: are they like a struct?

•OK, I'm seeing how strings work, but I'm still not in a place to make them work in my class. What's missing?? Take some time here to try to bridge the gap and break down the problem.

•I think part of it is figuring out the datatypes of the vars in my class → is it like a struct? an array of arrays? Or are they just one-off variables? How can I explore to find out? I suppose I could play around w/ intermediaries, or mess around w/ my supporting functions to make them work better. Maybe I should take a step back and explore structs more? Long story short, I definitely feel like I'm missing something @ I really need here.

•Perhaps I can convert my sandbox code into a simple struct just to play around w/ the idea a little? Not a bad idea.

2017/07/10 Monday

whew! Am I ready for this? A slug brain and a tense jaw trying to get into the swing of things?? This is my first actual week on my new schedule: MT, ThF, w/ W off. Will it work out? Time will tell! I hope so!

•OK, playing w/ struct + string to try to see if I can find a missing piece. Then, depending on what I learn, trying to convert that over to classes!

•hmm... I'm suspecting that while I could figure this out, I'm far enough away w/ few enough insights @ perhaps I should just leave this alone for now. Sure, I guess I could build a super-simple class up from the ground... and maybe OS the way to go... Gah!! My brain is sludge! So much sludge!

-What would my next step be to turn my struct into a class?

•I suppose the next step is to break up my file into multiple pieces, and then make a Makefile. Will everything still work??

*Makefile

•OK, given the state of things and my own psychology, it seems like it might be best to end things here. I could continue to tinker (by, for instance, removing 'string' vars from separated code), but I think I've already done @ and it wouldn't answer my current question anyways... So what to do in the 10+15 minutes remaining? Good question! Not much time to be practical on my web side (although I think I should work on @ during the next time segment) → but what else could I do? I suppose I could read up on/refresh my memory on pointers... Always a useful enterprise. ~~Ptt~~

>Ask Jim what's the deal w/ separate file errors

>Also ask what datatype my vars w/in Class are (array of arrays? solo vars? what?)

•OK, I had told myself that after my break I'd do web work. Should That I was only reading up on pointers & linked list classes as a filler for a short time segment. Is @ still the case? It probably should be, even though it would be fun to continue diving into classes. That's a good problem →

to have, I guess: it means I'm enjoying myself. ☺

>>> moving forward, looking ahead:
- make 5-1_Class_Car_A.cpp work!
- consider cleaning up Strings_a.cpp to be a little sleeker:
  - eliminate 'using namespace std' → consider manually typing std::string, etc, etc
    std::cout
- work on next step of ch 5 exercises → what's ~~he got more~~ none has he got for me to do?
  > more robust simple class
  > linked list, pointers, dependent classes and functions!

- wow, that's a lot! I think I'll be occupied for awhile there!

2017/07/13 Thursday

~~ah~~ cool. I've got my strings to work out for exercise 5-1. I suppose at this point I should read ahead to see what other exercises I can work out. ☺

Exercise 5-2

✓ create supporting function ⊕ returns a complete description of auto object as a formatted string → "1957 Chevy Impala"

```
std::string output_func(x,y,z){
    ⋮
    return output_string;
}
int main(){
    ⋮
    cout << output_string;
    return 0;
}
```

✓ Create second support method ⊕ returns the age of the auto in years
int age = (now - ~~get~~ get_year); → get tips from Jim, figure it out! ☺

2017/07/14 FRIDAY

- web app: Jim and I made some good progress on thinking through ways to move forward, but it seems to me that there's still plenty of heavy mental lifting to do there. Why not start my Friday morning w/ something a little more laid back, like C++ classes? Nice, self-contained C++ classes... ☺

5-3 exercise

✓ variable-length string functions from ch 4: ✓append, ✓concatenate, ✓characterAt
→ use them to create class for variable-length strings
→ make sure to implement all needed ✓constructors, a ✓destructor, and an overloaded assignment operator.

- ooh boy, ~~linked lists~~ arrays and heap pointers! Def. going back to ch 4 for a refresher. Also, what's the jist w/ 'overloaded' assignment operators? I saw those during online research of classes (I think) and had no idea what they were talking about either.

- holy cow! You can have a struct within a class?? Mind blown! *pew*
- boo -- I was hoping to be able to skip this final tutorial section (for now), but it turns out ☹ that's kind of the whole point to learn the required lessons for parts of exercise 5-3. That's ok: more learning!
  - Jesus --> deep copy functions somehow have something to do w/ the concept 'overloading assignment operator'. A) O.a.O. doesn't make sense on its own (to me), and as I'm trying to glean clarification and understanding, I find myself waist-deep in a bunch of confusing shit! Where have I wandered off to?? What's even going on?? OK -- deep breath -- then move forward. I can always say to myself ⊕ this biz has gotten too messed up for now, and I can pivot to something else. I could return to Ch 4 to play around w/ class-less linked lists (I definitely dropped the ball on ~~pra~~ taking the time to practice enough w/ those). Heck! I could even play more w/ plain pointers! Always a good concept to work on. And, of course, I could always pivot back to the web app. Busting my brain on 'cycles' and 'new cycle' functionality and design, or possibly finding another avenue to explore if ⊕ seems like too much for now (fixing CSS interactions w/ chart? creating data validation steps? Cleaning up cycle_brackets()? Simply inserting a pretty picture in the background??). There's Always more to do and explore, and it's important for me to remember that!
  (Boy, my smoothie is sweet! Perhaps too much...)

- operator overloading: feature of C++ ⊕ allows us to change what the builtin operators do w/ certain types.
  - > In this case, we want to overload assignment operator (=)
    - ↳ instead of default shallow copy, '=' calls our copiedList() method to perform a deep copy.

- operator overloading allows you to use an operator not as it was initially designed: say, by adding two class objects together instead of the more conventional adding two numbers together. But why the overload shortcut?? Why not just create an 'addition' function ☹ adds the two class objects together? Since they're not traditional numbers, ☹ should make sense, right → why doesn't ☹ seem to be the conventional practice, the best practice? Why instead do classes seem to rely on ~~to~~ operator overload as a matter of practice? I can understand ☺ a+b looks straightforward (vs contrasted against a.add(b); or something) → but ☺ appearance is deceptive since ☺ nice and shiny '+' operator functionality required ~~lol~~ some behind-the-scenes re-configuring, and could be confusing to read → my ☺ expectation for what '+' can do may be violated if I see it being overloaded all the time...

  ~~by~~ operator overload '=' → x = y ⇒ x.operator=(y); (function type and name; passed argument)

  === CLOSING THOUGHTS ===
  whew! it's apparent ☹ I've got lots to do and a long way to go to feel like I've really figured this class thing (and many of its dependent bits) out. But I'm having fun, I think I'm moving forward, and I'm learning lots! Here's hoping ☺ this weekend is fun and refreshing, and ☺ I'm ready to jump back in on Monday!

the majority of folks who consume animal products do so mindlessly, without consideration to their actual or optimal needs, or to the greater global cost. ~~In org~~ Assuming @ you value the lives and existence and sacredness of animals around you, there could be an argument made for still continuing said / some animal products if its done with high levels of care and consideration towards ~~the~~ Solving / thinking about the equation of greater global cost vs your optimal living / thriving quotient. To present the idea more practically, if you discover @ a specific level of, say, fish or organ meats consumption provides optimum health value, then perhaps @'s an acceptable solution / middle point? You should also research other supplements (vegan omega 3's, joint-friendly ~~soft~~ sources, high, ~~dose~~ bioavailable sources of vitamins and minerals -- along w/ how much / what ratios of @ you even need) → but if you determine @ your ~~psy~~ physiology / medical research / available supplement / food resources / your lifestyle / activity levels / etc etc → if you determine @ all of @ big equation works better / more optimally by consuming, say, fish, organs, and broth (at, perhaps, limited or ~~tightly~~ tightly monitored levels) then perhaps it's worth considering against the relative ethical / moral / environmental costs. Whew!! What a complex web of thought I'm trying to navigate !!! Where does it all leave me? At this point, I'd honestly say I'm not entirely sure. :)

"scanned in musings"

2017/07/25 Tuesday

OK! Starting a new day and a new project: back to C++ Classes! So... where was I before? Something about pointers... (oy! operator overloading!) (building pointer struct/class from scratch...) (holy shit, it's been 11 days since I last worked on this?! So long!)

- OK, good → I've found my old work and have looked up some notes. But, uh, I still have no idea what the hell I was trying to do! I suppose I should find my time-log. →→ that seemed like a whole lot of not helpful. I guess I get to craft my own destiny, then.

- OK → at this point, I suspect I'd be happy to create a variable-length string class ☺ simply cout << the resulting text, w/o concatenating or appending or performing any functions like that. Simple enough...?? I guess it's a place to start, and I can ramp up or down the complexity as I go.

- whew! there are just enough layers here ☺ I may need more direct tutoring/hand-holding. Waiting for help from Jimbo I imagine.

2017/07/27 Thur

lines

~~27~~ 28, 29, 32 → all will be encapsulated in functions (allocate, assign, deallocate).

===

2:30	6:15	5:10	7:10
2:40	-5:15	+2:00	-5:10
5:10	1:00	7:10	
5:10		~~1:15 deficit~~	

- I could work 1:00 ☺ to make my day's goal today: 6:15 hours worked
- Any extra time worked would go towards my weekly deficit (1:15).
- If I worked an extra 30-45min, I'd be halfway to ☺ goal, and would be working till 7:00 or 7:15.
- Kinda late, but since my schedules shifted back by 1 hour anyways, it seems a bit later than it actually is (by old ~~stards~~ standards, I'm done by 6:00 or 6:15, which would mean I'd have till 8:00pm (2hr) to chill, prepare, eat food --before winding down for the day) → shift all ☺ back by an hour, and I've got a food, chill window from 7:15 till 9:00.
- ~~OK, I'll put 2 hours in → working from 5:45~~ 5:45 till 7:45 ☺
- OK, I'll work from 5:15 till 7:15, which would add ~~45~~ min to the deficit created on Monday. I would have liked to have worked the whole 1:15 off by now, but I'm doing what I can and that's all I've got!

['a', 'b', 'c', 'd', ∅]
0 1 2 3 4
1 2 3 4 5

total 5:30 - 7:30
5:30 - 6:30
to fulfill today's goal
6:30 - 7:30
to make up for mom

l = ∅
a != ∅, l = 1
b != ∅, l = 2
c != ∅, l = 3
d != ∅, l = 4
∅ == ∅, l does nothing

l = 4
l = 4+1
return l (5)

original-length = 5
append_array = 5+1 (6)
copy arrays

['a', 'b', 'c', 'd', ∅, ␣]
0 1 2 3 4 5 6
1 2 3 4 5 6

append_array[5] = 1

2017/07/28 FRI

✓length() to ~~alloc~~ allocation_length() △ name for clarity?

— -6:00
script vs. 'bare metal' c++/compile

- Monday: explore and play w/ operator overloading to finish up requirements of 5-3 and move forward!

2017/07/31 Monday (∴ last day of July → sad for some reason)
the website is just zany enough right now ☺ I think I'll play w/ C++ first off this morning! OK! operator overloading... goodie...

```
public:
String & operator +(const String & rhs);

String & String::operator +(const String & rhs) {
    char_array = concatenate(rhs.char_array);      ??
    return *char_array; // (return *this;) ??
}
```

change concatenate() to return value of char* (not void):

```
char* String::concatenate(char* char_array_b) {
    ...
    ...
    delete [] char_array;
    char_array = cat_char_array;
    return *char_array;      ??
}
```

main

```
Std::cout << Word.get_String() * Second_char << Std::endl;
```

- I must admit, I'm scared to implement this code b/c I'm fairly confident ☺ it'll break, and then I'll be tempted to fix it but I won't know how so it'll just be a big mess and jumble. I could just plug in what I have, on the 5% chance ☺ it actually works → but then revert to an earlier place and then pivot to some easier operator overload code and try to slowly build up? Hmm... Or, since I know I have things to learn either way, I could just begin at the 'simpler' side and build from there?
- This is so dumb.

→ also, after this I should maybe go backwards and play w/ more simple pointers more. I really should practice by ☺ more!!!

rainbow belt
green curry
- dnd calendar
- how to spend time during work breaks to keep brain sane & best use time, energy

exercise 4-4 (going backwards)

- cout works by outputting strings ~~or characters~~ until there's a 0 at the end of the array.
- my output func would spy work like this

```
void output(char* s) {
    ~~for~~ s_length = length(s);
    ~~for~~ (int i=0; i<s_length; i++) {
```

- pretty simple method

```
    s_length = s[1] - 1;
    for (int i=1; i<s_length; i++) {
        cout << s[i];
    }
}
```

- reverse() → I do want to pass it the address of my_string (&s)

JIM pointers

value stored in memory @ refers to another place in memory

char*

MEMORY — n. 1 2 3 4 . . . . . 1001 1002 1003 1004 . . .

2 ↑ address of char*s valued stored 1002

1002 ↑ address of the value it points to

value it points to: 'a' (stored @ 1002)

- & s = 2
  s = &1002
- * s = 'a' (located @ location 1002)

(layer of indirection)

~~address of my_string = $50000~~
~~value of my_string =~~

~~my_string = 15~~
~~*my_string = 50000~~

- after assignment
  - * my_string always is the same as my_string[0]
  - naked my_string = 50'000

- & my_string = 15 (memory address of var)
- my_string = 50'000 (memory address @ var points to)
- *my_string = mystring[0] = 'a' (value of location in memory)

pass &my_string through append() -- ~~&~~ pass 15 through append() -- because within the function, I'm going to change where it (&my_string, @15) points to -- from ~~is~~ 50'000 memory location, to, say, 52'000 memory location.

~~→~~ [*s = new s] in append() → messes up main's stack, not append's stack

(terminal: if ctrl+s, then ctrl+q to reset!)

2017|08|01 TUESDAY

- when I have literal translation it works fine
- but when I try to reverse it goes all screwy.
- What's the deal?
  - → well, one of the deals is that I can't populate an empty array in reverse! ~~b[5]=b[5]~~ b[5] = a[1] doesn't work if b's an empty array!
  - → ANOTHER deal is that I can't iterate downward through j when it keeps getting redefined w|in the for() loop!
  - → AND, if j = 5, then it's never going to touch on any of the elements of the array, b|c their highest index is 4!
- → But when I fix all those problems, it works just fine! Woah! Science! Now to consider how to increment this up one level of complexity so that I can get closer to my final goal!

- hmm, fiddled around a bit w/ the heap|stack variables of reverse(). When I do things exactly like Jim showed me yesterday I can make it work beautifully. But any attempt at variation causes a disaster! I suppose diagramming would help, as would a refresher on array memory allocation...
- (boy, I'm tired. I'm a bit surprised at how tired I am... I guess the blanket showed me I'm more behind on sleep than I thought.)

- I suppose before I move onto structs, I should play more w/ what I have. Boy, memory allocation is confusing! In a way, I feel like I shouldn't worry so much, b|c real-world jobs may not do much in C++. but I suppose a boss would recognize C++ proficiency/familiarity for what it's worth (presumably at least something → as a deeper knowledge of nitty gritty computer work) → but I also want smarts in more 'modern' languages like javascript and ruby and others...

- But it's going to be a while before Jim's awake and ready to help me. What to do in the meantime? I could reread the pointer docs... I could and probably should attempt to diagram various styles of memory (heap, stack, etc) → being able to define exactly what I think I've got would do a lot to cement in what I do know and highlight what it is that I don't know. Perhaps the pointer chapter of tlap could also help as a supporting resource.

===

- while meditating today I thought about C++ and coding a little while my mind wandered during breathing - I thought that was kinda cool, actually → it suggests to parts of my brain are beginning to get reformed and reshaped.

body fawn doesn't stand a chance
body wolf > body fawn.

```
int main() {
    int static_array[5] = {1, 2, 3, 4, 5};
    int * dynamic_array = new int[5];
    dynamic_array[0] = 10; d_a[1] = 11; d_a[2] = 12; d_a[3] = 13; d_a[4] = 14; d_a[5] = 15;
}
```

- both are pointers, but one's static (on stack) while the other's dynamic (on the heap). But what does ⊛ actually mean for memory allocation?

```
& static_array    // memory address 520
& static_array[0] //  "      "    520
& static_array[1] //  "      "    524
  "      "[2] //  "      "    528
  "      "[3] //  "      "    52c (hexadecimal, remember)
  "      "[4] //  "      "    530
```

w/ Jim

it's nonsense for stack references to point up the stack, but not to point down.
(the high-up memory may be erased or filled w/ something different)
pass-by-reference: hand a stack reference up, which points down.

→ so fascinating to me that this level of stack-heap / where-am-I-at-on-either/both → → that all of that goes into play w/ all C++ functions and interactions. That level of knowledge about memory plays directly into feet-on-the-ground programming. Fascinating! And it shows me that I've still got a long way to go for all of this to become 'easy'. (plus my eyeballs hurt! Chill outside during next break).

===

STATIC ARRAY POINTER

1	2	3	4	5
0	1	2	3	4

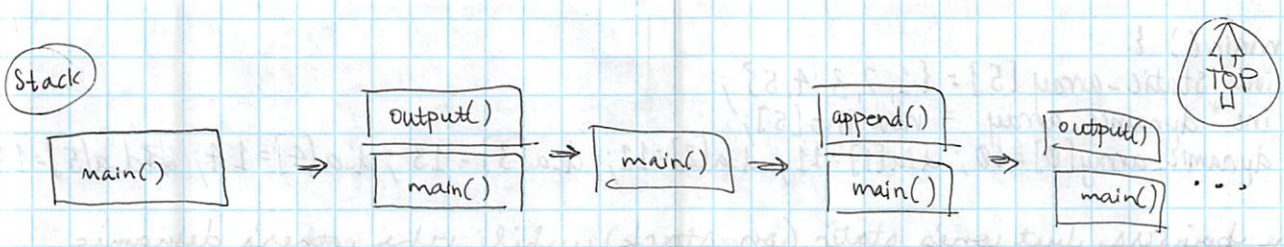
→ int static_array[5] = {1, 2, 3, 4, 5};

- ~~variable static_array → 0x7fff228dc230~~
- ~~address of static_array → 0x7fff228dc230 (on local stack)~~
- ~~address of static_array[0] →~~

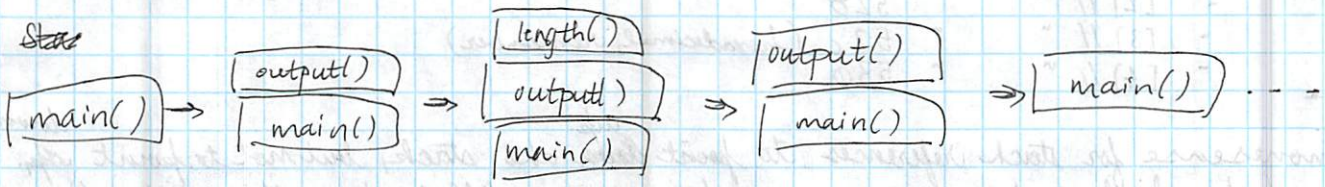
(value of)	
static_array	0x7ffc5d92c090
address of static_array	" "
address of static_array[0]	" "
" " [1]	" 094
" " [2]	" 098
" " [3]	" 09c
" " [4]	" 0a0
value pointed at by (static_array + 0)	1
" " " (" + 1)	2
static_array[0]	1
static_array[1]	2

===

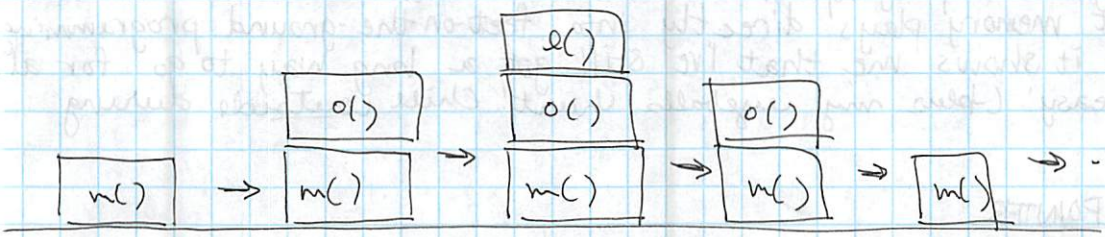
* address of dynamic_array and address of static_array are very close in stack memory.
* addresses of heap variables in a very different spot than stack memory addresses.



- because each of my functions are self-contained and don't call their own helper functions, my stack doesn't get @ deep. But if I had, for instance, a helper function length() which would be called to find the length of an array, the stack might look like:



- and... for clarity: I keep showing the stack bobbing up and down, with main() going all over the place. Actually, it's like:



- And, of course, this is all an abstraction of what's actually going on, but hopefully it makes some sense of actual reality...

DYNAMIC ARRAY POINTER

```
int *dynamic_array = new int[5];
dynamic_array[0] = 11; d_a[1] = 12; d_a[2] = 13; d_a[3] = 14; d_a[4] = 15;
```

11	12	13	14	15
0	1	2	3	4

(value of)	
dynamic_array	0x5585f591ec20
address of dynamic_array	0x7ffc5d92c088
address of dynamic_array[0]	0x5585f591ec20
" " [1]	" c24
" " [2]	" c28
" " [3]	" c2c
" " [4]	" c30
value pointed at by (dynamic_array + 0)	11
" " " + 1]	~~11~~ 12
dynamic_array[0]	~~12~~ 11
dynamic_array[1]	12

1 2 3 4 5 6
[6 | lucky]
0 1 2 3 4 5

= = = = = = = = = = = =

- ok → I want to play w/ this pointer stuff some more. what other function could I make for my array of chars? My dynamically allocated string?
  - (EASY) ✓ Copy → copy values of one array to another location in memory of same size.
  - Compare → look at two arrays and verify whether the contents are the same or not. → all the printable characters are the same

l	u	c	k	∅			
l	u	c	k	∅			

→ these are both the 'same' ←

5	l	u	c	k			
5	l	u	c	k			

* → copy function: I should create a check condition to make sure the copying-to array is >= than the original array

- Compare func → should I try to compare different layers of memory? If so, how does that look?
  → first, let me try to compare two stack variables.

✓ Shit, apparently first (and modified) version(s) of compare() actually doesn't work: if there's a difference in value of element of array, it still prints out ✓ it all checks out...
- hm... it doesn't make a comparison on the first pass of if(){} → both when there's an else statement and when there isn't one.
  → ha! I got too ~~Col~~ cocky / vag-ey w/ my placement of 'true' statement → It's all better now.

- how to compare a stack pointer w/ a heap pointer? Perhaps, for initial simplicity, I'll do arrays of intigers? Oy! Mayle not -- I'll do it if I need to.

```
char* my_string = new char[6];
my_string = ... "lucky";

Char Stack_compare[4] = { 4, 'a', 'b', 'c' };
```

- If I pass my_string & and stack_compare in compare() what's going on? my_string value is heap memory. stack_compare value is stack memory. Perhaps if I pass ~~&~~ & my_string to compare(), and then added extra layers in compare() to compensate for the added layers?

Struct: size, string (2 variables) ??

Class: dictionary (hash) key:value
linked list boxing champ (w/ random choice, wins
prints out rankings, outcomes)

oof! make the mistake of asking Jim what a dictionary is → big mistake..

2017/08/03 Thursday

I could move forward in a couple of directions, according to Tuesday's discussion. I could make a Struct of two components (one, an int ⊕ stores the length of a string, and another, an array ⊕ holds the chars of a string). And then (I think), I could just convert my current functions over to work like what I've got now w/ 4-4.
Jim was talking about something -- something about the differences btwn struct and class functionality. He definitely seemed to think it was important to tell me as I gain understanding. Something about the struct not being quite as robust as a class in a particular way, but ⊕ not necessarily being bad, per ~~se~~, just a different understanding and functionality. ... I think... → (Jim seemed to be leading on ⊕ struct's a good way to ease back into classes... but I want to stay on pointers a little longer)

I could also do a linked list. That's definitely something I want to play w/ more, too, and it may give me a chance to play more w/ pointers ⊕ aren't arrays. Jim, off the cuff, thought of this boxing game that could use linked lists: ~~once~~ spin randomly through the list to select a boxer, and then have the player challenge the boxer above him (ie, behind him on the linked list). ~~A die would~~ A coin toss would decide who won, and if the underdog won, then ~~the~~ his/her hed move up the rankings accordingly. The game would also print out who won the match, what the rankings are, etc.
This spontaneous idea definitely seems more interesting than tracking student grades or product inventory. ☺ And it would be kind of fun to create a dynamic, usable program -- to make it more alive. But it would also be more work.
Should I ~~the~~ try something other than boxing? A rap battle? Break dance-off? ☺ Or even just a ~~S~~ race → clear-cut rankings and winners, but much less violent?
OK → that's simple. I'll do 200 m dashes instead of boxing.

What should the linked list store? Name of runner, ranking, outcome of most recent race, and link to next node.

- std::string name
- int ranking
- int outcome_seconds
- (pointer to next node)

- There would be some user input → to initiate a race. So far that's it -- everything else is automated.
- Perhaps add new players into the mix???

highschool 200m dash
boy 20.41 s } cheetah-fast
girl 23.07 s }
(≈ iq = 160-170: ⊗ high-level)

median time:
boy: 25.64 s
girl: 30.27 s
(≈ iq = 101)

top 10%:
boy: 23.55 s
girl: 27.32 s
(≈ iq = 120)

- maybe also stats: average runtime, total # of runners, difference in time btwn first and last, etc
- Perhaps the user can select which runners compete against eachother (instead of being random)

• the 'ranking' variable may be redundant → the order in the list implicitly declares that fact, and it can be checked by evaluating the outcome_seconds variable. So I think I'll remove it.

[next goal: insert a node.]

```
void function( placeholder, new_node) {
if (placeholder == NULL)
if ( placeholder != NULL) {
    if ( placeholder -> next == NULL) {
        placeholder -> next =
```

(new_node -> next = NULL)

```
void function (placeholder, new_name, new_time) {
    run_node * new_node;
    when (placeholder -> next == NULL) {
    run_node * new_node;
    new_node -> name = new_name;
    new_node -> outcome_seconds = new_time;
    when (placeholder -> next == NULL) {
        placeholder -> next = new_node;
    }
    new_node = NULL;
}
```

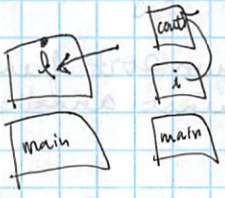
• classic problem. I need to find a way to keep track and advance one further. How do I do that? ... Uh... I thought I had an idea, but actually removed one more instead of adding one more. I know there's a way to have counters and all that, but I know that it works differently than arrays (no indexes) and ☹ I struggled w/ this last time (part of the reason I gave up, I think).

```
void add_new_at_front(run_node* existing_node,
void add_new_at_front(placeholder, std::string new_name, float new_time) {
    run_node * conductor;
    conductor -> name = new_name;
    conductor -> time_sec = new_time;
    conductor -> next = existing_node;

    existing_node = conductor;
    conductor = NULL;
}
```

Jim: deallocate node

• before I add new in the middle, figure out how to print a name in the middle.
• I feel like I need a counter() helper function




```
void counter(run_node* existing_node) {
    int counter = 0;
    while (existing_node -> next != NULL) {
        counter++;
    }
    counter++;
    return counter;
}
```

```
void insert_after_name(run_node ** existing_node, std::string insert_after_this_name, std::new_name,
        float new_time) {
    run_node * conductor = new run_node;
    conductor -> name = new_name; conductor -> time_sec = new_time;
    while ( * existing_node -> name != insert_after_this_name) {
    while ( * existing_node -> name != insert_after_this_name) {
        * existing_node = * existing_node -> next;
        run_node * next_node_down_the_list = * existing_node -> next;
```

(2017/08/04 FRI NOTES / THOUGHTS / DIAGRAM)

Jim C++ (gdb debugger)

~~15:27~~

```
while (run_200m -> next != NULL) {
    run_200m = run_200m -> next;    // traverse the list
}
```

placeholder = run_200m;

- first pass: ((placeholder = ~~run_200m~~ placeholder -> next) at beginning of while?)
  (run_200m = "Bianca") (run_200m -> next = "Akmal") . . . so:
  run_200m = run_200m -> next ==> run_200m = "Akmal"
  placeholder = ~~run_200m~~ placeholder -> next ==> placeholder = "Maurice"
- second pass:
  (run_200m = "Akmal") (run_200m -> next = "Maurice") . . . so
  run_200m = run_200m -> next ==> run_200m = "Maurice"
  placeholder = ~~run_200m~~ placeholder -> next ==> placeholder = NULL
- third pass:
  (run_200m = "Maurice") (run_200m -> next = NULL) . . . so:
  - ~~end of while loop. I could execute something here maybe?~~?
  - ~~set a placeholder to equal this ending place in time~~?
  - #) -> end of while loop. Can't do anything else here.

placeholder -> next = ~~new~~ conductor ~~Dana~~
run_200m = placeholder;
placeholder = conductor = NULL;

= = =

- (more ~~thoughts~~ on ~~qsort~~ qsort):
  - what else should I sort?
  - array of objects? wait -> I'm imagining javascript syntax w/ @ . . .
  - two-dimensional array?
  - or, maybe, I should play more w/ TLAP: Creating my own sort function and modifying values? Compare my array against another one's values? Let's read and explore!
  - Oh yeah, also play w/ std::sort() (not just qsort())
  - and, of course (~~too~~) ~~the~~ sorting exercises at end of chapter 3. TLAP.

= = = = (after Jim) = = = =

- qsort is more generic: uses pointers and allocates new places in memory as placeholder while I'm figuring shit out and sorting it. Insert_sort uses less memory b/c it's directly accessing and sorting the data. Jim went into much more detail and I didn't follow all of it, but there you go, @s what I've got now, and maybe I could play more later!

I think I'm sad I couldn't make Epic work. But, instead, I should focus on the cool new adventures I get to go on ~~in~~ in the near future in Lincoln! And my Epic experience is framing my ~~work~~ work and goals!

```
*existing_node->name = conductor->name;
*existing_node->time_sec = conductor->time_sec;   maybe don't need conductor
** ?? *existing_node->next = next_node_down_the_list;   find a way to look forward...

   *existing_n
   conductor = NULL;
   next_node_down_the_list = NULL;
}
```

- I think I'm close but I maybe need more placeholders so I don't break my base list?

```
void insert_after_name(run_node** existing_node, std::string insert_after_this_name,
    std::string new_name, float new_time) {
  run_node* placeholder = new run_node, *existing_node;

  while(*existing_node->name != insert_after_this_name) {
    placeholder = *
  while (*placeholder->name == insert_after_this_name) {
    (placeholder = placeholder->next;)  maybe remove? //don't remove: moves
    run_node* next_node_down_the_list = placeholder->next; //store following node
    placeholder->name = new_name; placeholder->time_sec = new_time;
    placeholder->next = next_node_down_the_list;
  }
  *existing_node = placeholder;
  placeholder = NULL;
}
```

Swap two steps

2017/08/04 FRIDAY (more of today's notes on back of pg ⑧)

yay, friday! although... that means a busy weekend ahead. ohh...

- ask Jim to help w/ adding new node at end.
- also ask him about memory of linked list → why do I need to deallocate list heads w/in helper functions? why can't I just do ⓐ w/ main list head in main() (why does it seem ⓐ I need to deallocate the whole list?)
- Man, I've got 10 minutes left in this time segment and I have no idea what to do → I feel stumped on my current path. I could try to think of what I could do for next time chunk?
- I could take a step back and write a sorting function for an array. I ~~know~~ think I'll need to sort my linked list to help my time_sec variables in order. I could play w/ the idea on an array to get the ball rolling, maybe...

→ hmm, ~~I've never~~ I guess I remember trying to do qsort() via TLAP and really struggling. Maybe I should go back to what to try to resolidify ⓐ lesson now ⓐ I've got a bit more patience?

→ wow, what a difference a little sleep fog can create!! These problems have such a different flavor today than they did yesterday!

Jim

- Another study path I could pursue in C++: GDB: The GNU Project Debugger (or perhaps other versions) → anyways, playing w/ debugger →

tools to help me out of sticky situations and to add to my skillset. Also, in Javascript, there are built-in browser tools to make this easier. I can also insert cout << statements everywhere to try to find ~~other~~ where bugs are: to report the state of things while the code is executing (if it ever gets @ far I guess...).

- Next week: maybe keep on w/ C++ for a bit: it's fun! Tackle the final leg of linked_list: add new node to middle (and maybe continue to study the code @ adds to the end. Also deallocate @ memory and ask Jim about deallocating w/ ~~functions~~ helper and linked lists and all @.
- I could probably keep going, but since it's Friday and it's gorgeous out and my brain is toast, maybe I'll call it a day.

2017/07/08 Monday

```
void insert_new_after_name(run_node ** existing_node, std::string ~~name~~ insert_after_this_name,
  std::string new_name, float new_time) {
    run_node * new_node = new run_node;
    run_node * placeholder; //more later
    run_node * one_node_ahead; //more later

    new_node->name = new_name; new_node->time_sec = new_time;

    if (*existing_node == NULL) {
      new_node->next = NULL;
      *existing_node = new_node;
    } else {
        placeholder = *existing_node;
        one_node_ahead = *existing_node->next;

        while (placeholder->name != insert_after_this_name) {
            placeholder = placeholder->next;
            one_node_ahead = one_node_ahead->next;
        }
        placeholder->next = new_node;
        new_node->next = one_node_ahead;
    }

    new_node = placeholder = one_node_ahead = NULL;
}
```

have an earlier check. figure this out too?

(|| while placeholder->next != NULL)

```
while (placeholder->name != insert_after_this_name) {
    placeholder = placeholder->next;
}
if (placeholder->name == insert_after_this_name) {
    //do above
} else {
    //do add_new_at_end().
}
```

• what else could I work on? Of course, I've got all sorts of questions for Jim re linked list: figuring out my conditions for insert-new-after_name; asking about deallocating memory; etc. I was also working on sort() functions last week. Tinkering w/ qsort and insertion_sort. Perhaps learn more about qsort vs insertion-sort and how they interact w/ memory? Are there online resources @ make @ comparison? I could also try to write another of my own sort function → something other than bubble. Or I could even improve insertion_sort so @ it doesn't bubble up. I could learn about other sort functions and try to implement those.

Or, moving away from sort and C++ altogether, I could go back to my web app. Where did I leave things there and what are my goals to move forward?

• Woah, it's been a full week since I touched that.

• Oh yeah, I was playing w/ end date stuff, but I decided to leave @ alone for a while. But I was also playing w/ moment.js and timeseries and making date gaps show up on my chart. and I do think I want it functionality/feature set. And I was going to maybe play w/ db queries to try to get the correct information pulled from there to make it easier to port relevant data to charts. But what exactly did I need?

- Total number of days in the chart: needs to be dynamic so @ it works for incomplete chart too.
- single object w/ {x: " ", y: " "} → object pairs. Pull it together somehow using query and assignments.
- Other stuff I'm sure but my brain's not remembering them all.

• One nice thing about Javascript/Nodejs is @ I can play and experiment on the terminal before I commit real code. Yay!

• Oh yeah, re linked list → I could SORT!

w/ Jim

```
int
    return -1 } if name's found
           0  } or isn't
           1  }
```

===

• ~~test~~ ~~[illegible]~~ • consider breaking linked list file apart for easier navigation and coding moving forward.

TUESDAY'S IDEAS:

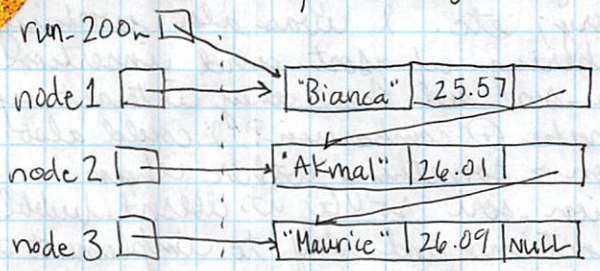
✓ deallocate memory in the heap! (big project w/c I've got all sorts)

• consider other linked list projects: sort; or I could sort other types of variables. Or I could look over my game's "rules" and see what else I could do to move forward w/ making my ~~or~~ current setup look more like my game.

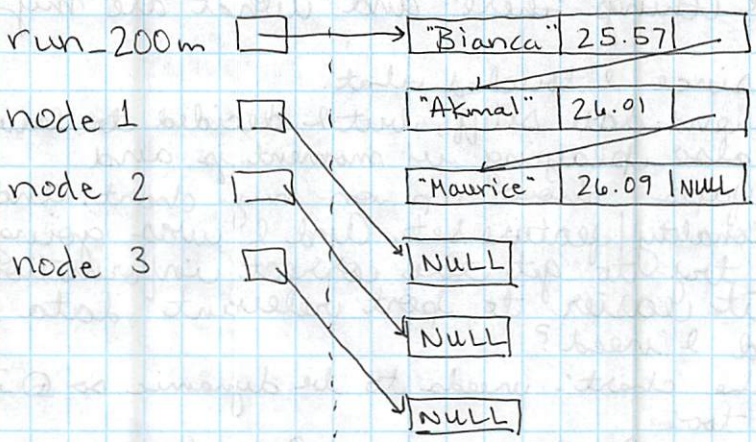
• ALSO web app: continue w/ gaps in chart and res.render logic. Oof!

2017/08/08 TUESDAY

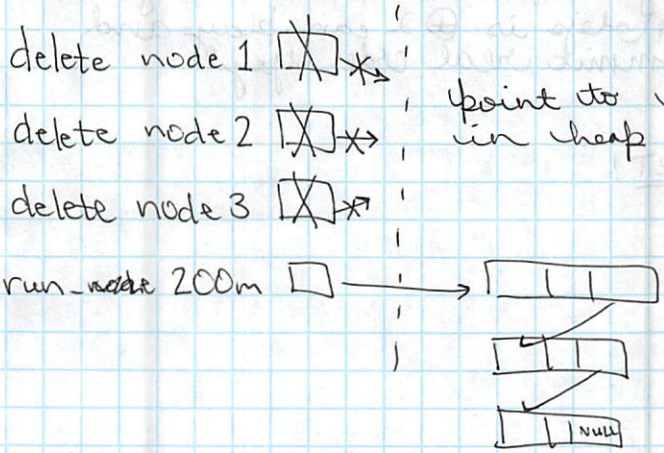
• deallocate heap memory!



• I need to set nodes 1,2,3 to something else before I deallocate their memory, otherwise deleting node 1,2,3 would also affect the memory & run_200m points to.



• set nodes 1,2,3 to null before deallocating their heap memory

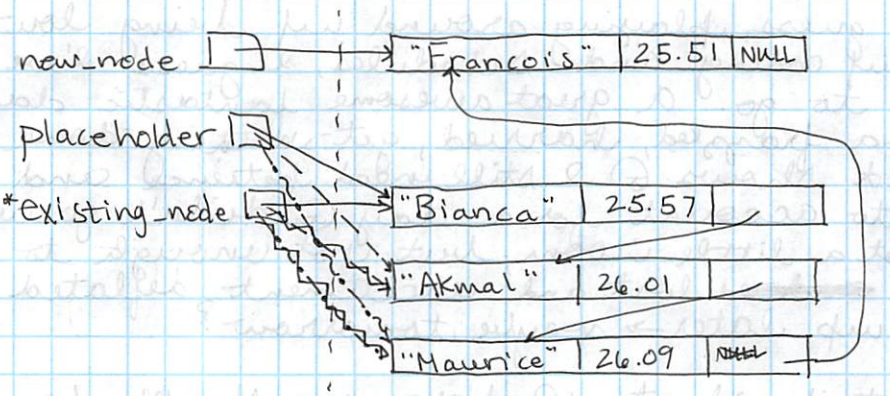


point to nothing in heap memory.

• run_200m still points to heap memory. I need to figure out how to deallocate @ memory too.

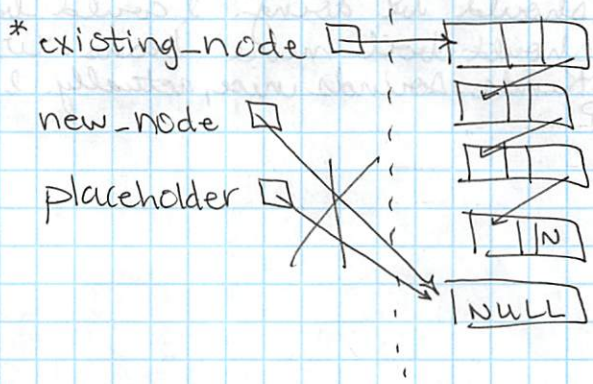
• but perhaps first I'll update all helper functions ~~to~~ to deallocate one-off nodes (whose contents are copied over to run_200m.

• but also, I should diagram more "complicated" helper nodes/lists, b/c -- since they only point to memory also allocated to run_200m, those placemarkers may actually be pretty simple to deallocate. Run_200m may be the only "complicated" list to deallocate from the heap.



~~12 more min~~

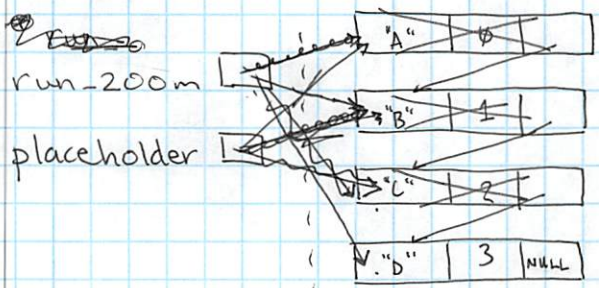
~~5:28~~



(delete)

- placeholder, conductor, etc etc, ino different from new_node → just need to make sure to set them to NULL before deallocating them!

~~new_node~~
~~placeholder~~ || - ok, I'm pretty sure I've addressed all heap memory other than @ pointed to by run_200m. ~~It~~ It'll require a little thought to determine how to do a deep deallocation!




```
~~run_node placeholder = new run_node~~
~~for~~
while (run_200m->next != NULL) {
    placeholder = run_200m;
    run_200m = run_200m->next;
    delete placeholder;
}
~~placed~~
run_node placeholder = new *run_node;
placeholder = run_200m;
~~placeho~~
delete placeholder;
delete run_200m;
```


- Jesu Cristo! What the fuck, dude?! I'm barely holding ~~things~~ ~~together~~ my coding and meditating and absolute bare minimum exercise, and then I get a bunch of extra junk to process. And none of it is urgent, some of it I can effectively ignore, but even being introduced creates ~~bad~~ clutter. And I kind of just can't handle it. My brain feels wrung out, and now I've completed my rote, mindless tasks, I kind of don't know how to move forward. I don't have the accessible bandwidth in my current state to think critically and notice little fiddly bits. Which is important b/c losing those brain capacities makes my coding and crafting efforts ~~100%~~ way more difficult or even impossible now.

And, while I'm, I guess, playing around w/ being low-key and kinda casual w/ coding and schedules, I guess I've still got a long ways to go. A great awesome fantastic day can be followed by a frazzled, harried, get-nothing-done day, and @ sucks. It shows @ I still need patience and ~~wiss~~ wiggle room to account for a day @ didn't go well. Thankfully I've got a little room, but not enough to make up for time ~~and so~~ lost and excitement deflated: I'll have to make it up later → maybe tomorrow?

> So, tomorrow. Strip sheets. Recharge myself. Maybe consider a fun project I could do in Ruby. Exercise is always on my list of things I should be doing. I could bake some yummy bread-like thing? I should boil more beans. What kind? Maybe lentils? w/ Rice? Kinda sounds nice, actually. I could maybe make a lentil loaf?

↑↑ SCANNED ↑↑