Deb's D&D for P3

dev and debug (with help from Perry Kivolowitz April 2012)
Friday 10/22/21 2:30-3:45pm
270 Soils

TODO:

1. vim: set configuration options, favorite vi commands
2. tree p3
    a. make libheap.so from myHeap.o
    b. make and run heap allocator test programs
    c. edit - make - run - repeat
    d. functions: provided and must be implemented
3. debugging: (slides based on Perry Kivolowitz's talk -- April 2012)
    a. The Scientific Method and defensive programming
4. divide and conquer: (gdb) GNU debugger
    a. launch
    b. basic operations
5. In Conclusion

my vi settings

vim ~/.vimrc

https://piazza.com/class/kt4m3f1whqj24w?cid=47

```
syntax on

set number

set ruler

set tabstop=4

set shiftwidth=4

set softtabstop=4


:color desert
```

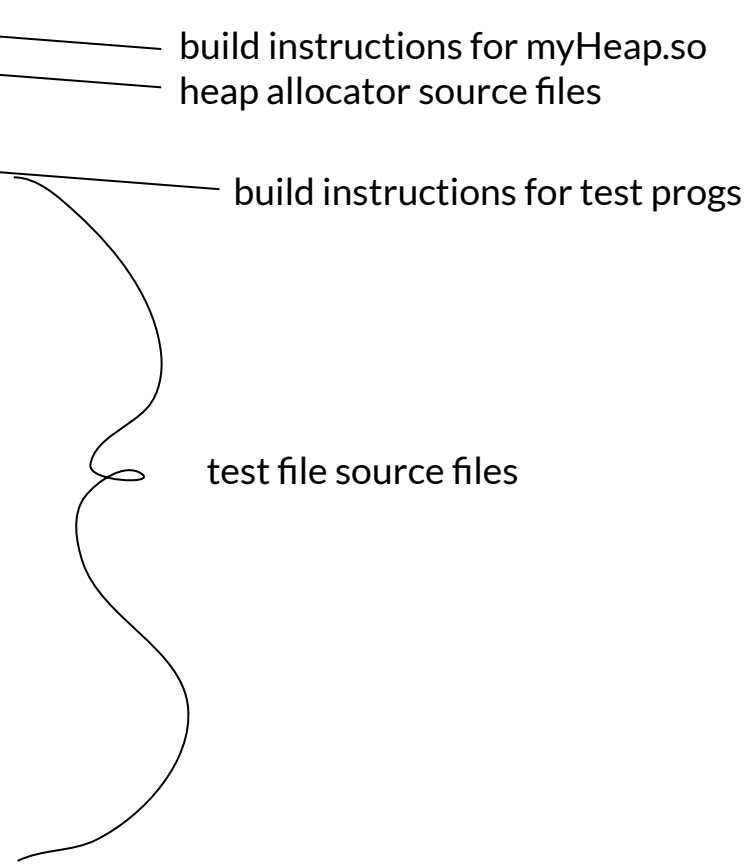
my favorite vi commands

```
:1,$s/[Tab]/     /g
```

```
/main
123G
yy p
r R
x p
dd D
cw dw
gg=G
ZZ
:tabe newfile
:n :p
```

```
tree p3
```

```
p3
├── Makefile
├── myHeap.c
├── myHeap.h
└── tests
    ├── Makefile
    ├── test_align1.c
    ├── test_align2.c
    ├── test_align3.c
    ├── test_alloc1.c
    ├── test_alloc1_nospace.c
    ├── test_alloc2.c
    ├── test_alloc2_nospace.c
    ├── test_alloc3.c
    ├── test_coalesce1.c
    ├── test_coalesce2.c
    ├── test_coalesce3.c
    ├── test_coalesce_deb1.c
    ├── test_coalesce_deb2.c
    ├── test_fit1.c
    ├── test_fit1_nospace.c
    ├── test_fit2.c
    ├── test_fit2_nospace.c
    ├── test_free1.c
    ├── test_free2.c
    └── test_writeable.c
```



p3 Heap Allocator

a. *Build myHeap.o and libheap.so*

```
cd p3
make
```

b. *Build executable test programs:*

```
cd p3/tests
make
./test_align1
```

Pro Tip: Open multiple terminal windows. One for p3 and the other for p3/tests
Pro Tip: Copy a test source to your own temporary mytest.c file and edit that to get started

_

edit
make
test
repeat

p3 Heap Allocator (Header)

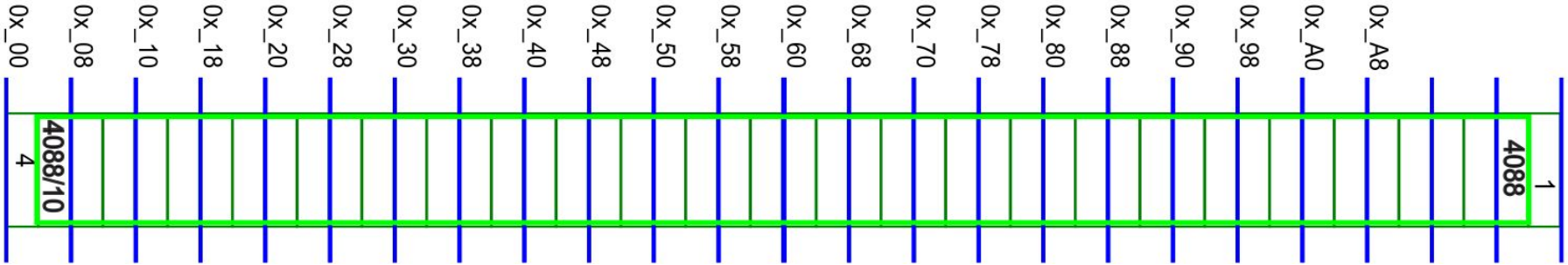
c. defined for you

```
int   myInit(int size);
void  dispMem();
```

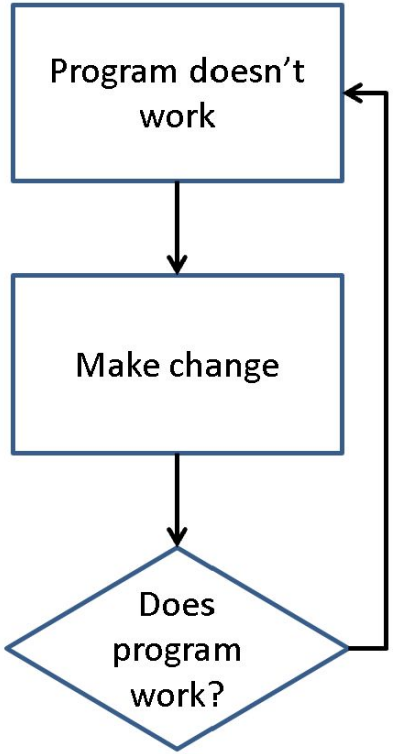
d. must define as described in comments

```
void* myAlloc(int size);
int   myFree(void *ptr);
int   coalesce();
```

Initial state of the heap after calling myInit(4096)



Common Debugging Algorithm



https://www.carthage.edu/live/profiles/1477-perry-kivolowitz

Perry Kivolowitz

Credit: Perry Kivolowitz April 2012 Debugging Talk

Kivolowitz Corollary

A fix is not a fix until you completely understand why it is a fix

Credit: Perry Kivolowitz April 2012 Debugging Talk

Defensive Programming

- if you “know” a condition is true, assert it
- this helps eliminate the impossible
  and identify the improbable

Credit: Perry Kivolowitz April 2012 Debugging Talk

Comment before CODE

Answer WHY instead of WHAT

- write down your thoughts to get clarity
- to help you recall what you were thinking ...

  a year from now? Next week? Tomorrow?

```
// increment j   // score_index
```

Credit: Perry Kivolowitz April 2012 Debugging Talk

Descriptive Variable Names

pk: What does **`ineedsleep`** do?

student: I don't remember, I was tired.

"keystrokes are guaranteed to end, debugging is not." pk

Credit: Perry Kivolowitz April 2012 Debugging Talk

Testing

Testing can only prove the presence of bugs, not their absence.

Edsger W. Dijkstra

Credit: Perry Kivolowitz April 2012 Debugging Talk

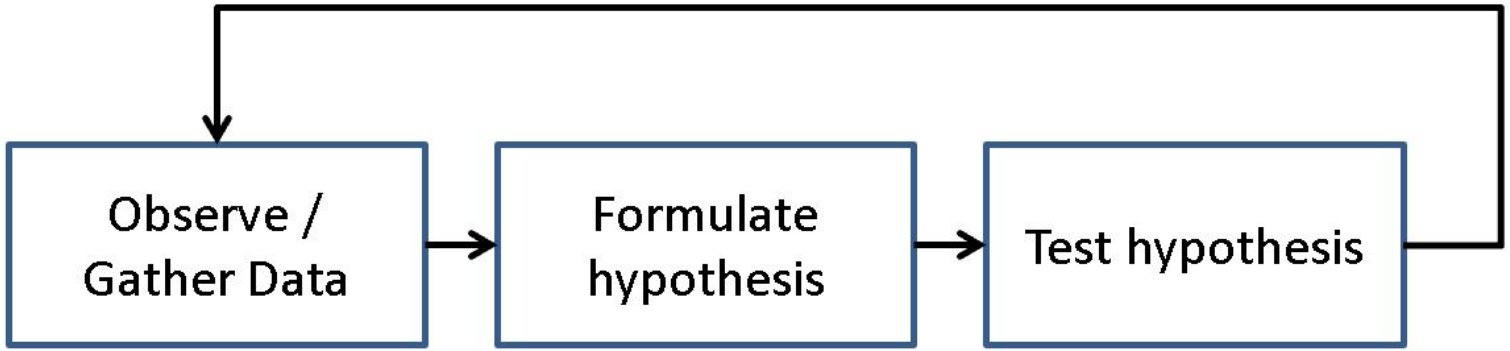
Quality Assurance (QA)

If you are fortunate to have a QA group

- It is not their job to find bugs
- That is your job
- It is their job to confirm bugs you found are gone

Credit: Perry Kivolowitz April 2012 Debugging Talk

Debugging Tool:
The Scientific Method



Credit: Perry Kivolowitz April 2012 Debugging Talk

Debugging Tools: test harnesses and incremental development

Write a minimal test harness

that manifests the bug

“Write in small units – Test in small units”

Credit: Perry Kivolowitz April 2012 Debugging Talk

Debugging Tool: debugger

- Single step
- Set Breakpoints
  - Always
  - Conditional
    - In debugger
    - In code
- Call stack

- Re-execution of code
- Immediate modes
- Value inspection
- Value modification

Credit: Perry Kivolowitz April 2012 Debugging Talk

GNU debugger

```
rockhopper-04$ gcc -g test.c
rockhopper-04$ gdb a.out
```

(gdb) **quit**

(gdb) **help**

(gdb) **run arg1 arg2**

```
rockhopper-04$ gdb mytest
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and
redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources
online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to
"word"...
Reading symbols from mytest...
(gdb)
```

Now What?

- set breakpoints
- run until next breakpoint is reached
- step through code or continue to next **bp**
- print variables and expressions
- edit - compile - run (in gdb)
- repeat until works as advertised

gdb commands

```
> gcc -g ... -o test
> gdb test.c
(gdb) break main
(gdb) run
(gdb) print argc
(gdb) help display
(gdb) display
(gdb) step
(gdb) next
(gdb) continue
(gdb) quit
```

```
q quit
h help
r run arg1 arg2
b break main
b myAlloc
r run
s step
n next
p prev
c continue
d display
p print
```

Solve the Puzzle: Divide and Conquer

Where are the values for the output set and modified?

Work forward from declaration and initialization toward ...

Work backwards from the output ...

Split distance in half and check the value.

Repeat until instruction that sets or computes value incorrectly is found.

Fix it.

In Conclusion

Bugs want to be found

- Listen, hypothesize, test

Hone your craft and

- master the art of debugging

Deb: Many thanks to Perry Kivolowitz who taught me to use Science when debugging my code!

Credit: Perry Kivolowitz April 2012 Debugging Talk