

this is week 3

fall 2013

playlist50

Royals (Lorde) | Love Somebody (Maroon 5)
| Safe and Sound (Capital Cities)

agenda

resources

GDB

sorting & searching

resources

lecture notes & source code

cs50.net/shorts

study.cs50.net

man

Google

cs50.net/discuss

OHs

me!

Tell Jason what you think about his teaching - both positive and constructive comments welcome!

Be honest and sincere, you'll stay anonymous:

Describe Jason Hirschhorn's good or bad qualities here -- this will help him/her to develop.

☐ **Recommended:** Allow Jason Hirschhorn to respond privately. You'll stay anonymous.

Say it.



Jason Hirschhorn has got 5 anonymous opinions

Get your feedback URL - 20 second sign-up

Your full name

Your password

Your feedback URL

sayat.me/

Sign Up

<http://sayat.me/cs50>

GDB

squash those bugs!

get started

```
jharvard@appliance (~/.pset3) gdb <executable>
```

```
jharvard@appliance (~/.pset3) gdb ./caesar
```

useful commands

break (b) main

print (p)

run <arguments>

info locals

next (n)

continue (c)

step (s)

disable

list

quit (q)

your turn: caesar.c

Use GDB to find the bugs in
the program.


```
Terminal
File Edit View Terminal Tabs Help
jharvard@appliance (~/.Dropbox/cs50/week3): check50 2013.pset2.caesar caesar.c
:) caesar.c exists
:) caesar.c compiles
:) encrypts "a" as "b" using 1 as key
:) encrypts "barfoo" as "yxocll" using 23 as key
:( encrypts "BARF00" as "EDUIRR" using 3 as key
  \ expected output, but not "EAUIRR\n"
:) encrypts "BaRFoo" as "FeVJss" using 4 as key
:) encrypts "barfoo" as "onesbb" using 65 as key
:) encrypts "world, say hello!" as "iadxp, emk tqxxa!" using 12 as key
:) handles lack of argv[1]
https://sandbox.cs50.net/checks/9de716808c28448bb89f4b1309fa6540
jharvard@appliance (~/.Dropbox/cs50/week3): █
```

let's turn that frown upside down
(there may be more bugs as well!)

sorting & searching

4 8 15 16 23 42

your turn: binary.c

Implement an iterative version of binary search using the following function declaration:

```
bool binary_search(int value, int values[], int n);
```

Where `value` is what you are searching for in the `values` array of size `n`. No need to write a main function; simply include `helpers.h` in another `.c` file.

`your turn: binary.c`

And one more thing...please
start by writing some
pseudocode.

your turn: binary.c

```
while length of list > 0
```

```
    look at middle of list
```

```
        if number found, return true
```

```
    else if middle higher, search left
```

```
    else if middle lower, search right
```

```
return false
```

your turn: selection.c

Implement selection sort using the following function declaration:

```
void selection_sort(int values[], int n);
```

Where `value` is what you are searching for in the `values` array of size `n`. No need to write a main function; simply include `helpers.h` in another `.c` file.