4156

11/3/16

go over team testing assignment

ad for 6156

go over midterm

in-class team/pair exercise

5? min pair

5? min team

cross-team project comparison

↳ ask Ewan to dom class Tuesday

prepare for that class as assignment

4/56 11/3/16

debugging

1st step — overcome your belief that your code is right & the computer is wrong

0th step - Use a real debugger or at least assertions not print

compiler, interpreter, runtime environment, library, etc. bugs are very rare

what you told the computer to do was wrong

best case - your unit test detected the bug, so you know the bug manifest in that unit (the actual root cause of the bug may be elsewhere, a caller of the unit or other code that sets variables used by the unit)

worst case — your customer detected the bug

this usually means that none of your existing test cases detected it so need to reproduce (we'll assume we can)

4/56 11/3/16

finding the bug is a process of checking your beliefs about what the code is doing

you might believe that a certain variable has a certain value at a certain place in the program

you might believe the else branch of an if-then-else was the one executed

you might believe that when you call a function, it receives certain parameters with certain values

it is possible to use a debugger, or even just print, to check these beliefs
- but you can't check everything

crashing bug - run code in debugger which will automatically be triggered at point of crash

check values of all variables in scope at point of crash, do they meet your beliefs? do you know what your beliefs should be at this point?

4156 11/3/16

if you don't know what to expect at this statement, find other "nearby" statements where you do know — insert debug breakpoints or value assertions there

non-crashing bugs - start with breakpoint just before bug exhibited (e.g., output is wrong value), then same idea, check beliefs

narrow down where belief went wrong "binary search" between place where belief holds & place where belief doesn't hold (halfway point)

if you can find ~~a place~~ the earliest place where belief doesn't hold, the bug may be obvious — or not

if not, trace backwards to determine how the execution got to this point
- through control branches
- through function/method calls
- through "slices"

4156 11/3/16

so far, we've assumed a single value is wrong
what if it is a table/set/list of values?

try to find "smallest" set that exhibits
same (or equivalent) bug

"binary search" also applies here
replace half of set with known
good values, then other half
(or simply remove half)

example sets
words (a whitespaces) in word processor
HTML (or CSS, etc.) in browser
numbers in spreadsheet
# of shots available in shooting game

so far considered cases where something
is wrong with the code - could also
be due to *missing* code
(or existing code that is never called)

where should this code be & where
should it be called?