Grift 2: Grift Harder
Thoughts about Assignment 3
Thoughts about Assignment 3

- I noticed a few trends in how people were working
Thoughts about Assignment 3

• I noticed a few trends in how people were working
  ○ I’d like to address some of those trends
Thoughts about Assignment 3

• I noticed a few trends in how people were working
  ○ I’d like to address some of those trends
  ○ We’ll do that in a moment...
Thoughts about Assignment 3

• I noticed a few trends in how people were working
  ○ I’d like to address some of those trends
  ○ We’ll do that in a moment...

• The assignments are difficult
Thoughts about Assignment 3

• I noticed a few trends in how people were working
  ◦ I’d like to address some of those trends
  ◦ We’ll do that in a moment...

• The assignments are difficult
  ◦ Yet only three people came to my office hours last week
Thoughts about Assignment 3

• I noticed a few trends in how people were working
  ◦ I’d like to address some of those trends
  ◦ We’ll do that in a moment...

• The assignments are difficult
  ◦ Yet only three people came to my office hours last week
  ◦ I am not on campus most days and I am very difficult to reach on weekends
Thoughts about Assignment 3

- I noticed a few trends in how people were working
  - I’d like to address some of those trends
  - We’ll do that in a moment...
- The assignments are difficult
  - Yet only three people came to my office hours last week
  - I am not on campus most days and I am very difficult to reach on weekends
  - I try to respond to emails, but there can be a lag
Thoughts about Assignment 3

• I noticed a few trends in how people are working
  ○ I’d like to address some of those trends
  ○ We’ll do that in a moment...

• The assignments are difficult
  ○ Yet only three people came to my office hours last week
  ○ I am not on campus most days and I am very difficult to reach on weekends
  ○ I try to respond to emails, but there can be a lag
  ○ If you have a timely need office hours (mine of a TA’s) are your best bet
Cheating

• Don’t
Trend #1
Trend #1

- Know your operating system and relevant tools
Trend #1

• Know your operating system and relevant tools
• You should be able to do the following:
Trend #1

- Know your operating system and relevant tools
- You should be able to do the following:
  - Install software
Trend #1

• Know your operating system and relevant tools
• You should be able to do the following:
  ○ Install software
  ○ Run tools (like Dr. Racket or some git GUI)
Trend #1

• Know your operating system and relevant tools
• You should be able to do the following:
  ○ Install software
  ○ Run tools (like Dr. Racket or some git GUI)
  ○ Use CLI tools if necessary (like the Racket repl, or git)
Trend #2
Trend #2

• Many of you that needed help had a very common issue:
Trend #2

• Many of you that needed help had a very common issue:
  ○ You knew what needed doing but you’d trip yourself up
Trend #2

• Many of you that needed help had a very common issue:
  ◦ You knew what needed doing but you’d trip yourself up
  ◦ Structure, structure, structure
Trend #2

• Many of you that needed help had a very common issue:
  ○ You knew what needed doing but you’d trip yourself up
  ○ Structure, structure, structure
  ○ Languages provide abstractions for a reason: use them
Functions
Functions

• Structure of your code $\leftrightarrow$ structure of your problem
Functions

• Structure of your code ↔ structure of your problem

• Decompose the problem into smaller bits:
Functions

• Structure of your code $\leftrightarrow$ structure of your problem

• Decompose the problem into smaller bits:
  ○ This helps you *think*
Functions

• Structure of your code ↔ structure of your problem

• Decompose the problem into smaller bits:
  ○ This helps you *think*
  ○ This helps you *test*
Functions

- Structure of your code ↔ structure of your problem
- Decompose the problem into smaller bits:
  - This helps you *think*
  - This helps you *test*
- Some of you would misidentify the errors because you assumed too much!
Functions

• Structure of your code $\leftrightarrow$ structure of your problem

• Decompose the problem into smaller bits:
  ○ This helps you *think*
  ○ This helps you *test*

• Some of you would misidentify the errors because you assumed too much!
  ○ smaller functions $\Rightarrow$ fewer assumptions
Functions

• Structure of your code \(\leftrightarrow\) structure of your problem

• Decompose the problem into smaller bits:
  ○ This helps you *think*
  ○ This helps you *test*

• Some of you would misidentify the errors because you assumed too much!
  ○ smaller functions \(\Rightarrow\) fewer assumptions

• Let me show you an example.
Trend #3
Trend #3

• Asking good questions
Trend #3

- Asking good questions
- Screenshots are an inappropriate way to ask for help.
Trend #3

• Asking good questions

• Screenshots are an inappropriate way to ask for help.
  ◦ I do not have a language implementation in my head (I wish I did)
Trend #3

• Asking good questions

• Screenshots are an inappropriate way to ask for help.
  ○ I do not have a language implementation in my head (I wish I did)
  ○ When you share code, the very first thing I do it try to run it!
Trend #3

• Asking good questions

• Screenshots are an inappropriate way to ask for help.
  ○ I do not have a language implementation in my head (I wish I did)
  ○ When you share code, the very first thing I do is try to run it!

• Reporting an compiler/runtime/interpreter error without context make it difficult for anyone to tell you anything new
Trend #3

• Asking good questions

• Screenshots are an inappropriate way to ask for help.
  - I do not have a language implementation in my head (I wish I did)
  - When you share code, the very first thing I do it try to run it!

• Reporting an compiler/runtime/interpreter error without context make it difficult for anyone to tell you anything new
  - If you aren’t clear about what you’ve already tried it will be difficult to know how to help
Some thoughts on git

- It’s a powerful tool
Moving on
Stacks!
Stacks!

- We’ve already agreed that stacks are useful for managing runtime environments
Stacks!

• We’ve already agreed that stacks are useful for managing runtime environments

• For our compiled code, let’s use the \texttt{rsp} register to point to the \textit{base} of our stack
Stacks!

• We’ve already agreed that stacks are useful for managing runtime environments

• For our compiled code, let’s use the \texttt{rsp} register to point to the \textit{base} of our stack

• Consider the following
Stacks!

- We’ve already agreed that stacks are useful for managing runtime environments

- For our compiled code, let’s use the \texttt{rsp} register to point to the \textit{base} of our stack

- Consider the following

\begin{verbatim}
(let ((x 1)) (let ((y (add1 x))) y))
\end{verbatim}
• We’ve already agreed that stacks are useful for managing runtime environments

• For our compiled code, let’s use the rsp register to point to the base of our stack

• Consider the following

  \[(\text{let } ((x \ 1)) \ (\text{let } ((y \ (\text{add1} \ x))) \ y))\]

• Let’s walk through what’s happening to the stack in our compiled code
Stacks?
Stacks?

- Let’s look at a binary operator
Stacks?

• Let’s look at a binary operator

\[
(let ((y 5)) (+ y (add1 y)))
\]
Stacks?

• Let’s look at a binary operator

\[
(let \ ((y \ 5)) \ (+ \ y \ (add1 \ y)))
\]

• How do we manage the two arguments to +?
The Compiler
The Compiler

• We can’t do it naively, consider:
• We can’t do it naively, consider:

\[
\begin{align*}
\text{(define (compile-+ e0 e1 c)} \\
\text{ (let ((c0 (compile-e e0 c)))} \\
\text{ (c1 (compile-e e1 c)))} \\
\text{ `(@c0} \\
\text{ ,@c1} \\
\text{ (add rax ???))))}
\end{align*}
\]
The Compiler
The Compiler

• What are some alternatives?
The Compiler

- What are some alternatives?
- With those alternatives in mind, consider:
What are some alternatives?
With those alternatives in mind, consider:

\[ (+ \ (\text{add1} \ 2) \ (\text{add1} \ 3)) \]
The Compiler

• What are some alternatives?
• With those alternatives in mind, consider:

\[
(+ \ (\text{add1} \ 2 \ \text{add1} \ 3))
\]

\[
(+ \ (\text{add1} \ 2 \ 3))
\]
The Compiler

• What are some alternatives?

• With those alternatives in mind, consider:

\[
(+ \ (\text{add1} \ 2) \ (\text{add1} \ 3))
\]

\[
(+ \ (\text{add1} \ 2) \ 3)
\]

\[
(+ \ (\text{add1} \ 2) \ x)
\]
The Compiler
The Compiler

• Before we dive in, let’s add comments
The Compiler

• Before we dive in, let’s add comments
  ◦ You should all feel empowered to experiment
The Compiler

- Before we dive in, let’s add comments
  - You should all feel empowered to *experiment*
  - Reminder to José: in assembly they’re called `remarks`