Section 8: **Midterm Review**

Four Drill Questions Students Struggled With, Cheet Sheet Walk Through, and Mini-Midterm

CS 164 @ UC Berkeley, Fall 2021

For each expression below, will it result in a **compile-time** error, a **run-time** error, or a **valid** result when run in the **class compiler**?

A. (+ 1 3 4)
B. (+ 1 (let ((x 2)) false))
C. (left (left (pair 1 false)))
D. (if true (* 2 3) (** 2 3))

If our language supported **only booleans and numbers** (no pairs, vectors, refs, etc.) which of the following steps could we **elide** when calling a C function? Explain each.

A. Saving rdi to the stack
B. Bumping rsp to reflect the contents of the stack
C. Ensuring that rsp is a multiple of 16

How many 8-byte heap "cells" will be used when the following program is **compiled and executed**?

```
(let ((x false))
    (pair 1
        (if x
            (pair 2 (pair 3 (pair 4 false)))
            (pair 2 (pair 3 false)))))
```

The ocaml code below implements a **bool? operation** in the compiler. What should binary_number_1 be replaced with? Your answer should be a number in binary format (i.e., "0b" followed by some 0s and 1s).

```
| Lst [Sym "bool?"; arg] →
    compile_exp arg
    @ [And (Reg Rax, Imm binary_number_1); Cmp (Reg Rax, Imm binary_number_2)]
    @ zf_to_bool
```

Go Through Cheat Sheet

Bonus: Mini Midterm

(Disclaimer: Not Created By Prof. Chasins)

For each expression below, will it result in a **compile-time** error, a **run-time** error, or a **valid** result when run in the **class compiler**?

A.
```
(let ((v1 (vector 1 true)))
  (let ((v2 (vector 1 true)))
    (vector-get v1 1)))
```
B. (vector-length (pair 1 2))
C. (list? (pair 1 2))
D. (vector-get (if (not (char? #\a)) (vector 1 true) (vector 2 true)) 1)
E. (num->char 0)

The OCaml code below implements a **left** projection operation in the compiler. Write **three** s-expressions of the form **(left ...)**. One that will result in a **run time** error, one that will result in a **compile time** error, and one that will result in the **number 22**.

```
| Lst [Sym "left"; e] ->
    compile_exp tab stack_index e
    @ ensure_pair (Reg Rax)
    @ [Mov (Reg Rax, MemOffset (Reg Rax, Imm (-pair_tag)))]
```

What does the **interpreter symbol table** look like at each point in the code?

```
(let ((z false) (x 10)
      do (A_EXP)
            (pair 1
                  (if z
                        (let (x (+ x 1)) (let (y 9) B_EXP))
                        (let (y 8) (C_EXP))
                  )
            (x + 1)
            (D_EXP)))
```

A_EXP: _{z:false; x:10}_____ C_EXP: _____

B_EXP: _____ D_EXP: _____

What does the **compiler symbol table** look like?
(var stack index in order of appearance)

```
(let ((z false) (x 10)
      do (A_EXP)
            (pair 1
                  (if z
                        (let (x (+ x 1)) (let (y 9) B_EXP))
                        (let (y 8) (C_EXP))
                  )
            (x + 1)
            (D_EXP)))
```

A_EXP: __{z:1; x:2}_____ C_EXP: _____

B_EXP: _____ D_EXP: _____