Worksheet 6B

Quiz 1: *Semantics*

What should the following programs *evaluate* to?

Program	Result
`nil`	
`(vec (+ 5 5) (+ 10 10))`	
`(let (p (vec 10 20)) (+ (head p) (tail p)))`	

`(fun (sum l)`
`  (let (r 0)`
`  (loop`
`  (if (= l nil)`
`  (break r)`
`  (block (set! r (+ r (head l)))`
`  (set! l (tail l)))))))`
 	
`(sum (vec 10 (vec 20 (vec 30 nil))))`	

Quiz 2: *Abstract Syntax*

Write the *abstract syntax* for our language.

```
enum Expr { // ...




}
```

```
enum Index {




}
```

Quiz 3: *Evaluator Value*

How to extend `Val` and `eval` to support `vec`?

```
fn eval(e, env, funs) -> Val {
  match e {
    Expr::Nil          => { ____________________________ }

    Expr::Vec(e1, e2) => { ____________________________

                           ____________________________

                           ____________________________ }

    Expr::Get(e, i) =>   { ____________________________

                           ____________________________

                           ____________________________ }
}
```

Quiz 4: *Assembly*

Let's write the assembly code for evaluating `vec` and `vec-get`!

Program	Assembly
`nil`	
`(vec 10 100)`	
`(vec e1 e2)`	
`(let (p (vec 10 100)) (vec-get p 0))`	`; <(vec 10 100)>`
`(let (p (vec 10 100)) (vec-get p 1))`	`; <(vec 10 100)>`
`(vec-get e idx)`	`; <e>`

Quiz 5: *Your turn!*

What is something you found confusing in today's lecture (or earlier)?