Structured data!

```
expr := ... | (vec <expr> <expr>)
            | (vec-get <expr> 0) | (vec-get <expr> 1)
```

```
(vec 10 20)
```

```
(let (p (vec 10 20))
  (+ (vec-get p 0) (vec-get p 1)))
```

```
(defn (head l) (vec-get l 0))

(defn (tail l) (vec-get l 1))

(defn (inc xs)
 (if (= xs nil)
   nil
   (vec (+ (head l) 1) (inc (tail l)))))

(inc (vec 10 (vec 20 nil)))
```

```
(defn (sum lst)
 (let (total 0)
   (loop
     (if (= lst nil) (break total)
       (block
         (set! total (+ total (head lst)))
         (set! lst (tail lst)))))))

(sum (vec 1 (vec 2 (vec 3 nil))))
```

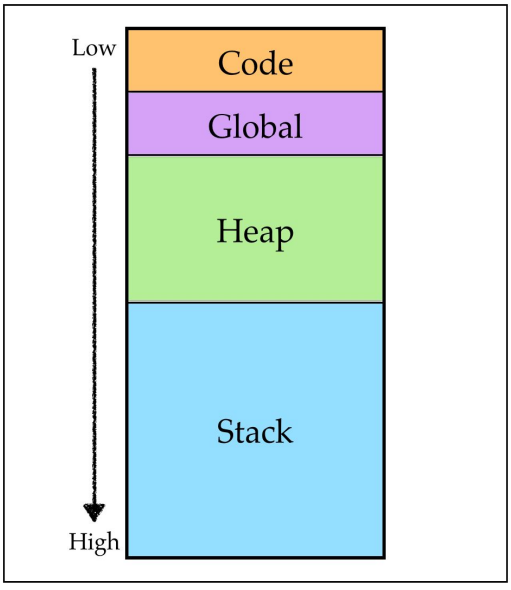
```
expr := ... | (vec <expr> <expr>)
            | (vec-get <expr> 0) | (vec-get <expr> 1)
```

```
enum Expr {
    ...
    Vec(                    ),
    Get(                    )
}
```

```
enum Index {


}
```

Where to *store* a `(vec 10 20)` ? on stack?



How to represent pointers?

number	xxx0
true	0111
false	0011
pointer	x001