Tutorial 2

- Quick review of C Syntax
- Translating Racket to C
- Implementing a ceiling function
- Recursion in C

Sample C Program

```c
#include "cs136.h"

// sum_first_squares(n): calculates the sum
// 1^2 + 2^2 + 3^2 + ... n^2
// Requires: n >= 0
int sum_first_squares(int n) {
    assert(n >= 0);
    if (n) {
        return n * n + sum_first_squares(n - 1);
    } else {
        return 0;
    }
}

int main(void) {
    printf("sfs(10) = %d
", sum_first_squares(10));
}
```

The previous program illustrates many C syntax elements.

- `#include "cs136.h"` (ignore for now)
- `purpose` statement, `requires` in contract
- use of `{}`, `()`, indentation, `;`
- static `typing`: `int x`
- any non-zero value is “true”
- `assert`, `if`, `return`
- `int main(void)`
- `printf` syntax with "%d"
My first C programs

All we have so far is:

- int types
- simple functions
- main with only `trace_int`

Let's try using them in seashell

Racket Translation

Translate the following functions into C:

```racket
(define (min a b)
  (cond [(<= a b) a]
    [else b]))

(define (min4 a b c d)
  (min (min a b) (min c d)))

(define (sum-digits n)
  (cond [(< n 10) n]
    [else (+ (remainder n 10)
       (sum-digits (quotient n 10)))]))
```

Integer Division

Define the following C function:

```c
// ceiling(a,b): produces the value of a/b, rounded up to
// the next largest integer
// Requires: a >= 0, b > 0
```
Recursion

Define the following C function: (use recursion)

//add_n(n): Produces the sum of the sequence
//from 1 ... n
//Requires: n>=0