Tutorial 2

- Quick review of C Syntax
- Translating Racket to C
- Implementing a ceiling function
- Recursion in 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\n", 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"
Racket Translation

Translate the following functions into C:

```
(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:

// ceiling(a,b) produces the value of a/b, rounded up to
// the next largest integer
// requires: a >= 0, b > 0
Recursion with side effects

Define the following C function: (use recursion)

```c
// fizzbuzz(n, fizz, buzz) produces a sequence
//     from 1 ... n with the following replacements:
//     replace numbers with divisible by fizz with fizz
//     replace numbers divisible by buzz with buzz
//     replace numbers divisible by both with fizzbuzz
//     requires: 1 <= fizz, buzz, n

For example:

fizzbuzz(16, 3, 5) produces the output:

1 2 fizz 4 buzz fizz buzz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz 16.

(note the spacing and the final period)