Week-7

CMPE-100 Classwork

Hint: Recursion

Repetition, or doing a series of similar computations, is accomplished in Racket by using recursion. A function is recursive if it is applied within its own body expression. The following function computes *n!*, or the product of all integers from 1 to *n* (where *n*is positive).

```
(define (fact n)
  (cond
    [(= n 1) 1]
    [else (* n (fact (- n 1)))]))

> (fact 10)
3628800
> (fact 50)
30414093201713378043612608166064768844377641568960512000000000
000
```

The evaluation of `(fact n)` for large `n` requires space (memory) roughly proportional to `n`, because the computation of `(fact n)` requires work to be done after the recursive application `(fact (- n 1))` (namely the multiplication by `n`). We can avoid this space overhead by using *tail recursion* (intuitively, ensuring that the recursive application is the "last thing done").

```
(define (fact-helper n acc)
  (cond
    [(= n 1) acc]
    [else (fact-helper (- n 1) (* n acc))]))
(define (fact n) (fact-helper n 1))

> (fact 10)
3628800
```

Racket will evaluate an application of this version of `fact` in constant space (apart from the space required for `acc`, which is accumulating the result), regardless of the value of the argument. However, due to the importance of recursion in Racket, the space allocated for managing control (the *stack*) is larger than in many imperative languages, so it is not necessary to write everything in a tail-recursive fashion.

Questions:

1. According to the hint, write a function that takes two integer n and m and calculates total of integers which are in the interval. If there is not an interval, programs gives error message.

   Example:

   (interval-sum 4 9) = 26

   (interval-sum 4 4) = “No Interval”

2. According to the hint, write a function that takes two integer m and n then calculates their greatest common divisor.

   Example:

   (biggest-divisor 12 8) = 4

   (biggest-divisor 54 24) = 6