CS115: Module 6 Extra Practice Problems

1. Write a function \((\text{find-multiples loi n})\) that consumes a list of integers \(loi\) and an integer \(n\) and produces a list of integers from \(loi\) that are multiples of \(n\).  
   Example: 
   \[
   (\text{find-multiples (list 0 3 4 -5 10 13 17 7 -12 20 25) 4}) \Rightarrow (\text{list 0 4 12 -12 20})
   \]

2. Write a function \((\text{keep-in-range loi a b})\) that consumes a list of integers \(loi\) and two integers \(a\) and \(b\) where \(a \leq b\) and produces a list of integers from \(loi\) between \(a\) and \(b\) (inclusive).  
   Example: 
   \[
   (\text{keep-in-range (list 10 13 4 5 -10 14 120 0 -13 -40 100) -11 50}) \Rightarrow (\text{list 10 13 4 5 -10 14 0})
   \]

3. Write a function \((\text{create2dlist n})\) that consumes a natural number \(n\) greater than zero and produces a list of lists (\(\text{list (list 1) (list 1 2) ... (list 1 2 ... n)}\)) starting from 1 up to \(n\).  
   Example: 
   \[
   (\text{create2dlist 4}) \Rightarrow (\text{list (list 1) (list 1 2) (list 1 2 3)(list 1 2 3 4)})
   \]

4. Write a function \((\text{create-list n k})\) that consumes a number \(n\) and a natural number \(k\) and produces a list of \(n\)’s repeated \(k\) times.  
   Example: 
   \[
   (\text{create-list -4 2}) \Rightarrow (\text{list -4 -4})
   \]

5. Write a function \((\text{mult a b})\) that produces the product between two natural numbers \(a\) and \(b\).  
   Note: You are not allowed to use \(*\) or \(/\).  
   Example: 
   \[
   (\text{mult 5 3}) \Rightarrow 15
   \]

6. Write a function \((\text{my-remainder a b})\) that produces the remainder when \(a\) is divided by \(b\).  
   Note: You are not allowed to use remainder or quotient.  
   \(a\) is a Nat and \(b\) is a positive Nat.  
   Example: 
   \[
   (\text{my-remainder 10 3}) \Rightarrow 1
   \]

7. Write a function \((\text{my-quotient a b})\) that produces the quotient when \(a\) is divided by \(b\).  
   Note: You are not allowed to use remainder or quotient.  
   \(a\) is a Nat and \(b\) is a positive Nat.  
   Example: 
   \[
   (\text{my-quotient 10 3}) \Rightarrow 3
   \]

8. Write a function \((\text{pow x n})\) that consumes a number \(x\) and a natural number \(n\) greater than zero and produces \(x^n\).  
   Example: 
   \[
   (\text{pow -1 3}) \Rightarrow -1
   \]
9. Write a function `(pow x i)` that consumes a number `x` and an integer `i` and produces \(x^i\).
   Example:
   `(pow 4 -2) => 0.0625`

10. Write a function `(calc lon)` that consumes a list of numbers `lon` and produces the same list of numbers squared `(calc (list v1 v2 ... vn)) => (list v1^2 v2^2 ... vn^2)`.
    Example:
    `(calc (list 2 6 5)) => (list 4 36 25)`

11. Write a function `(group-by-type s)` that consumes string `s` and produces a list with three strings where the first string only contains alphabets, the second strings only contains natural numbers, and the third string contains every other character in that string that are neither an alphabet nor a natural number. Note: You are not allowed to use string-append
    Example:
    `(group-by-type "67abc*1-8a") => (list "abca" "6718" "*-")`

12. Write a function `(alphabet-upto n)` that consumes a natural number `n` between 0 and 25 (inclusive), and produces the letters from the constant `alphabet` in position 0 to `n`.
    Useful constant
    `(define alphabet "abcdefghijklmnopqrstuvwxyz")`
    Examples:
    `(alphabet-upto 3) => "abcd"
    (alphabet-upto 15) => "abcdefghijklmnop"

13. Write a function `(my-range start end step)` that produces a list of integers from `start` to `end`, but not including `end`, stepping by `step`. See Module 6 Slide 20.
    Examples:
    `(my-range 4 7 1) => (list 4 5 6)
    (my-range 5 0 -1) => (list 5 4 3 2 1)`

14. Write a function `(lst-factors n)` that produces all the factors of `n`, starting with `n`.
    Example:
    `(lst-factors 100) => (list 100 50 25 20 10 5 4 2 1)`

15. Write a function `(ordered-factors n)` that produces all the factors of `n` in order from 1 to `n`.
    Example:
    `(ordered-factors 100) => (list 1 2 4 5 10 20 25 50 100)`

16. Write a function `(rev-prefixes s)` that behaves like the reverse of prefixes in Module 6 Slide 25. `rev-prefixes` should find all the prefixes of `s`, starting from the string itself to the shortest prefix (the empty string).
    Example:
    `(rev-prefixes "abc") => (list "abc" "ab" "a" "")"