# CSCE 476/876 Spring 2005
# Recitation exercises #1

Yaling Zheng
Constraint Systems Laboratory
University of Nebraska-Lincoln
yzheng@cse.unl.edu

January 25, 2005

---

Disclaimer: the content of this document includes material borrowed from AI and Lisp text books.

---

1. Define a function `compute-n4` that takes a number $n$ and returns $n^4$.

   If you put your code in a file named `week3.lisp`, then you can first load your code into the lisp environment by the following command:
   `(load "week3exer.lisp")` or `:ld week3exer.lisp`
   Then you compile the file using the following command:
   `(compile-file "week3exer.lisp")` or `:cl week3exer.lisp`

2. Define a function coverting temperature from Fahrenheit (F) to celsius (C) by the equation $C = \frac{F-32}{1.8}$.

3. Define a function that takes a list and returns the length of the list. Do not use the CL primitive `length`.

4. Define a function that takes a list and return the first three items and the last three items. For example, for the list `'(a b c this is a list 1 2 3)`, this function returns `'(a b c 1 2 3)`.

5. Given a list of lists, return the union of these lists. For example, for the list `'((1 2)(1 3)(1 5 6))`, this function returns `'(1 2 3 5 6)`. Do not use the CL primitive `union`.

6. Compute the summation of 1 through a specified positive integer.

7. Define a function, `count-letters`, that takes a list and returns the number of every distinct element in this list. Use a hash-table to store the result. For example, for the list `'(1 2 1 a b a c)`, this function returns a hash-table with the following items:

| *key* | *val* |
|---|---|
| 1 | 2 |
| 2 | 1 |
| $a$ | 2 |
| $b$ | 1 |
| $c$ | 1 |

8. Define a function, `count-letter2`, that takes a string and returns the number of every distinct letter in this string. Use a hash-table to store the result. For example, for the string `THIS IS A GOOD COURSE`, this function returns a hash table with the following items:

| *key* | *val* |
|---|---|
| *g* | 1 |
| *h* | 1 |
| *i* | 2 |
| *o* | 3 |
| *r* | 1 |
| *s* | 3 |
| *t* | 1 |
| *u* | 1 |
| *Space* | 4 |
| *a* | 1 |
| *c* | 1 |
| *d* | 1 |
| *e* | 1 |

9. Define a function, `reachable`, that takes three parameters: a list representing the edges of a directed graph, source vertex $u$, and destination vertex $v$. The function returns true if $u$ can reach $v$ and return false if $u$ cannot reach $v$.

   An example of a directed graph represented by edges are `'((u1 v1)(u1 v3)(v1 v4))`.

10. Define a predicate, `bipartite`, that determines whether or not an undirected graph is bipartite.