Fall 2019
CSCI/PHIL 4550/6550 Introduction to AI
Assignment 1: Learning Pseudocode

General Information

Deadline: 10:59 am Tuesday Sept. 3
Worth: 75 pts (undergraduates) 100 pts (graduates)

The Assignment

The purpose of this assignment is to make you think logically, learn, and practice writing solutions to problems in a stylized language such as *pseudocode*. Your grade will be based on the correctness of the pseudocode. Please be as clear and detailed as possible while writing the pseudocode. Furthermore, it is a good practice to include explanatory comments in your pseudocode.

Note: This assignment is not a group project and everybody should work on it individually. Students should not consult any previous solutions of questions in this assignment from any source.

In order to learn how to write in pseudocode, please see the Appendix B.2 of the textbook and the text file labeled "Pseudocode-Manual" in eLC course pages. Also, take a look at some examples written in pseudocode on page 47 (Fig. 2.7), page 49 (Fig. 2.10), and page 136 (Fig. 4.11) of the textbook.

Problems

1. (15 points) This is a two-part problem.

(a) Write a function MY-LIST-INTERSECTION that takes two lists as input and returns all elements that are common in the two lists. You need not worry about the lists containing embedded lists. For example,

   - *MY-LIST-INTERSECTION ([d e f g],[d g j])*
     output: d g
   -
   - *MY-LIST-INTERSECTION ([e d d f g g],[d d g j])*
     output: d g

(b) Write a function MY-LIST-UNION that takes two lists as input and returns all elements that are present in either lists without repeating any. You need not worry about the lists containing embedded lists. For example,

   - *MY-LIST-UNION ([d e f g],[d g j])*
     output: d e f g j
   -
   - *MY-LIST-UNION ([e d d f g g],[d d g j])*
     output: e d f g j

2. (25 points) This is a three-part problem.

   (a) Write a function ADDVEC that takes a single list of numbers as input argument and returns the sum of the numbers. Assume the input list will not have embedded lists. Return 0 if the list is empty.

   (b) Write a function VECMUL that will take as input two simple lists of numbers. VECMUL should multiply these lists coordinate-wise as one would multiply vectors. If one list is longer than the other, the result should be as if the shorter were padded with ones.
   For example: *VECMUL ([2 3 4 5],[1 4 5 2 14])*
   output: [2 12 20 10 14]

   (c) Use ADDVEC and VECMUL to define an inner product function INNPROD that multiplies the lists coordinate-wise and sums the resulting products.
   For example, *INNPROD ([1 2 3 4], [7 8 9])*
   output: 54

3. (15 points) Write a function OCCURRENCES that takes a list and indicates how many times each element appears in the list, sorted by most common to least common (break ties any way you like). Don't forget that elements of a list may be lists too! For example,

   - *OCCURRENCES ([a a b a c c d d])*
     output: a 3, c 2, d 2, b 1
   - *OCCURRENCES ([a a [b] a c c d [c [d]] [b] b d])*
     output: a 3, [b] 2, c 2, d 2, b 1, [c [d]] 1

4. (20 points) A *triangular number* counts the objects that are needed to form an equilateral triangle. Specifically, the n^{th} triangular number is the number of dots or balls composing a triangle with n dots or balls on a side, and is equal to, $T_n = \sum_{k=1}^{n} k = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$. Write a function MY-TRIANGLENOS that checks if the given list of positive integers is a sequence of triangular numbers. The lists will not have embedded lists. Look at the examples below:

   - *MY-TRIANGLENOS ([1 3 6 10 15 21])*
     output: true
   - *MY-TRIANGLENOS ([66 36 190])*
     output: true
   - *MY-TRIANGLENOS ([1 2 3])*
     output: false

   Note: The next two problems is for graduate students only

5. (10 points) Write the function SUBSETS that takes a list and returns the list of all subsets of the elements in the list. Note that the elements will not be lists themselves, and the list will not have any duplicate elements. E.g.,

   - *SUBSETS ([a b c])*
     output: [[] [a] [b] [c] [a b] [b c] [a c] [a b c]]

6. (15 points) A *square number* counts the objects that are needed to form a square. Specifically, the n^{th} square number is the number of dots or balls composing a square with n dots or balls on a side, and is equal to, $T_n = n^2$. A *square triangular number* is a number which is both

a triangular number and a perfect square (see Q4 for the definition of a triangular number). Write a function MY-SQTRINOS that checks if the given list of positive integers is a sequence of square triangular numbers. The lists will not have embedded lists. Look at the examples below:

- *MY-SQTRINOS ([1 36 1225 41616])*
  output: true
- *MY-SQTRINOS ([1 3 9 36])*
  output: false

What and how to hand it in

You'll upload the **typed** pseudocode for your functions. The document should include your name, student id, and all the function definitions. Please include explanatory comments in your pseudocode as much as possible.

Please submit your assignment by the deadline using eLC.

Assignments that are **late** but within a day of the deadline will be penalized 33% of the total number of points. Assignments submitted later than one day will not be accepted.