Computer Science CSC 324S: Programming Languages
University of Toronto – St. George Campus
Assignment PrologA
Due Date: Friday, 9 April, noon exactly

Because of the extension until Friday, no late days may be used on this assignment. Furthermore, note that the due date of noon is very firm – any assignments received even five minutes late will receive a zero.

There are two more questions coming later this week, but we haven't taught you everything you need to know yet. These questions will be relevant to the final exam. (As are all of the ones in this handout.)

All programs should be written in Prolog and should be well documented and thoroughly tested. Make sure you use good logic programming style; marks will be deducted for programs that look like translated Lisp, Turing or C.

Assignments should be handed in electronically, using the 'submit' command available on CDF machines. Here is how to use it for assignment prologB.

```
          submit -N  prologB  csc324h  x.P

where 'submit'     is the command to submit your assignment electronically,
      '-N prologB' indicates that you wish to create a subdirectory
       'prologB'   in your submit directory containing that assignment,
       'csc324h'   is this course, and 'x.P' is the file with your
                   submitted assignment.

If, for some reason, you decide later that you want to overwrite your
earlier submission (you cannot "unsubmit"), you must explicitly tell
submit to do so with the '-f' switch:

   submit  -N  prologB  -f  csc324h  x.P

For more information on how to use submit, type 'man submit' to access
manual documentation.
```

1. Write a predicate `sum_pairs` that is true if and only if, for each item in `L2`, if that item is in position i then that item is the sum of the numbers in positions i and $i+1$ in `L1`. Also, the last element of each must be the same.

Here is an example from a working program:

```
| ?- sum_pairs([1,2,1],[3,3,1]).
yes
| ?- sum_pairs([1,4,6,4,1],X).
X = [5,10,10,5,1]
yes
```

Hint: This has a three-line solution.

2. Pascal's triangle looks like this:

```
row 0              1
row 1            1   1
row 2          1   2   1
row 3        1   3   3   1
row 4      1   4   6   4   1
```


CONT'D...

Informally, each number is the sum of the two numbers above it.
This is how we index into the triangle:

```
                    P(0,0)

                P(1,0)  P(1,1)

            P(2,0)  P(2,1)  P(2,2)

        P(3,0)  P(3,1)  P(3,2)  P(3,3)
```

Row i looks like this:

P(i,0) P(i,1) ··· P(i,i)

The values in row i are defined recursively as follows: P(i,0) and P(i,i) are both 1, and for all $j, 0 < j < i$, P(i,j) = P($i-1$,$j-1$) + P($i-1$,j).

Write a predicate `row_match(L1,L2)` that is true if and only if `L1` is the row below `L2` in Pascal's triangle. (Ignore the problem of figuring out whether `L2` is, indeed, a row of the triangle.)

Here is an example from a working program:

```
| ?- row_match([1,2,1], [1,3,3,1]).
yes
| ?- row_match([1,4,6,4,1],X).
X = [1,5,10,10,5,1]
yes
```

Hints: You will find your answer to the previous question very useful. What is the difference between `row_match` and `sum_pairs`? This has a one-line solution.

3. Write a predicate `pascal(X,L)` that is true if and only if `L` is row `X` of Pascal's triangle.

Here is an example from a working program:

```
| ?- pascal(3,[1,3,3,1]).
yes
| ?- pascal(4,L).
L = [1,4,6,4,1]
yes
| ?- pascal(0,L).
L = [1]
yes
```

Hint: You will find your answer to the previous questions very useful. This has a two-line solution.

4. Write a predicate `subst(Element, New_value, List, Result)` that returns true if `Result` is `List` where all occurrences of `Element` at every level in `List` are replaced with `New_value`.

Here is an example from a working program:

```
| ?- subst(a,b,[[x],[a],[x]],Y).
Y = [[x],[b],[x]]
yes
| ?- subst(a,X,[a,b,c],[d,b,c]).
X = d
yes
```

Hint: You may need to use the `append` predicate.