ECS 140A: Summer 2024
Homework Assignment 4

Due Date: No later than Friday, July 26, 11:00pm

Submit your solutions via Gradescope as a single file named "hw4.pl". Solutions will be autograded, so use the procedure names specified below. You may need to write additional procedures other than the ones listed here to make things work – be sure to include those when you submit your solutions. (Of course, those additional procedures will use different names than the ones specified below.)

Grading for each problem will be on a 3-point scale for each solution
(4 problems x 3 points maximum per solution = 12 points maximum).

Problem 1: Write a procedure

```
shuffle(L1, L2, L3)
```

which returns true if list L3 is the result combining lists L1 and L2 such that the first element of L3 is the first element of L1, the second element of L3 is the first element of L2, the third element of L3 is the second element of L1, and so on. For example,

```
shuffle([a,b,c],[d,e,f],[a,d,b,e,c,f])
```

returns true. You may assume that L1 and L2 have the same number of elements. This should also work:

```
?- shuffle(X,Y,[1,2,3,4,5,6]).

X = [1, 3, 5],
Y = [2, 4, 6].
```

Problem 2: Write a procedure

```
double(L1, L2)
```

which returns true if every element in list L1 appears twice in L2, according to the pattern in the following example:

```
?- double([a,b,c],[a,a,b,b,c,c]).
true.
```

Note that the elements that are duplicated are also adjacent to each other.

These cases should also work:

```
?- double(X,[a,a,b,b,c,c]).
X = [a, b, c].

?- double([a,b,c],X).
X = [a, a, b, b, c, c].
```

This case should not work:

```
?- double([a,b,c],[a,b,c,a,b,c]).
false.
```

Problem 3: Write a procedure

```
no_duplicates(L1, L2)
```

which returns true if list L2 is the result of removing all duplicate elements from list L1. For example,

```
no_duplicates([a,b,c,b,d,b],[a,c,d,b])
```

returns true (note that the last duplicate 'b' is the one that remains). This should also work:

```
?- no_duplicates([a,b,c,b,d,b],X).
X = [a, c, d, b] ;
X = [a, c, d, b] ;
false.
```

Problem 4: Write a procedure

```
same_elements(L1, L2)
```

which returns true if lists L1 and L2 contain exactly the same elements, although possibly in different order. For example,

```
same_elements([a,b,c],[b,c,a])
same_elements([a,b,c],[a,c,b])
same_elements([a,b,c],[c,b,a])
```

all return true. This should also work:

```
?- same_elements([a,b,c],X).
X = [a, b, c] ;
X = [b, a, c] ;
X = [b, c, a] ;
X = [a, c, b] ;
X = [c, a, b] ;
X = [c, b, a] ;
false.
```