Exercise 1 (Operators): (6 points)

Please give definitions for operators such that Prolog can answer queries for the following Prolog program:

```
lst_christmas.
i gave you my heart.
%but the very next day.
you gave it away.
rudolph the red nosed reindeer had a very shiny nose.
we wish you a merry christmas and a happy new year.
```

You should define operators such that the following queries can be answered:

?- What christmas.
   What = lst

?- i gave you What.
   What = my heart

?- Who gave it away.
   Who = you

?- Who had a very shiny nose.
   Who = rudolph the red nosed reindeer

?- rudolph the red nosed reindeer had What.
   What = a very shiny nose

?- we wish you What and Whatnot.
   What = a merry christmas
   Whatnot = a happy new year
Exercise 2 (Operators): (5 points)

Please define the operators :=, ?, and : such that you can use the following predicate in Prolog:

\[ T_1 := B ? T_2 : T_3 \]

Here T1, T2, and T3 are Prolog terms for which you may assume that they do not contain any operators of a priority \( \geq 900 \). Then the above predicate is supposed to do the following: If the condition B holds, then the terms T1 and T2 are supposed to be unified, otherwise the terms T1 and T3 are supposed to be unified. You can test your implementation using the following example:

?- Z = 2, W = 1+Z, X+Y := Z < 0 ? 3+4 : 5+W.
Z = 2,
W = 1+2,
X = 5,
Y = 1+2.

Exercise 3 (Cuts): (2 points)

Consider the following implementation of the factorial function in Prolog via the predicate fact:

\[
\text{fact}(0,1).
\]
\[
\text{fact}(N,F) :- N1 \text{ is } N-1, \text{ fact}(N1,F1), F \text{ is } F1*N.
\]

The query \(?- \text{fact}(2,X).\) has the solution \(X = 2\). If one requests further solutions from Prolog (by pressing the key ','), one enters an infinite branch in the SLD tree, and the query does not terminate. Please give an alternative implementation for the factorial function via a predicate factcut/2 where you use cuts to make sure that no further solutions are searched for after a first solution has been found. You should make sure that any query \(?- \text{factcut}(t,X).\) for \(t \in \{0,1,2,\ldots\}\) terminates (i.e., the SLD tree for the query should be finite).

Exercise 4 (Cuts): (3 + 3 + 3 \times = 6 + 3^* \text{ points})

a) Consider the following Prolog program for division on natural numbers:

\[
\text{div}(X,0,Z) :- !, \text{ fail}.
\]
\[
\text{div}(0,Y,Z) :- !, Z = 0.
\]
\[
\text{div}(X,Y,Z) :- !, \text{ minus}(X,Y,U), \text{ div}(U,Y,Z).
\]
\[
\text{minus}(X,0,X).
\]
\[
\text{minus}(s(X),s(Y),Z) :- \text{ minus}(X,Y,Z).
\]

Please answer the following questions for all three cuts in this program and give short explanations for your answers.

- Is it a green or a red cut?
- How would computations with this program change if this cut was omitted?

b) Consider the following Prolog program:

\[
\text{appendex}(X,Y,Z) :- \text{ app}(X,Y,Z), !.
\]
\[
\text{app}([],L,L).
\]
\[
\text{app}([X|XS],L,[X|ZS]) :- \text{ app}(XS,L,ZS).
\]

Please give three terms t1, t2, t3 such that the query \(?- \text{appendex}(t1,t2,t3).\) yields an infinite SLD-tree.
c)* Consider the following Prolog program:

q1 :- ;((!,p(b)),p(a)).
q2 :- or((!,p(b)),p(a)).

or(X,Y) :- X.
or(X,Y) :- Y.

p(a).

The two queries ?- q1. and ?- q2. have different results. Please explain this difference.

*Hint: Use additional sources from the Prolog literature.*

Exercise 5* (Meta-Predicates): (3* points)

In Prolog the built-in predicate =/2 can be used to check whether two terms unify (without occurs check). However, after a successful proof of \( s = t \), the variables in \( s \) and \( t \) are instantiated by their most general unifier. Therefore, e.g. the following query will fail:

?- X = 1, length([], X).

The reason is that after showing \( X = 1 \), the variable \( X \) has been instantiated by 1 and cannot be instantiated by 0 any more, which would be required for satisfying length([], X). Here, length/2 is a pre-defined predicate to compute the length of a list.

Please write a predicate check_unify/2 that takes two terms as arguments such that check_unify(\( s \), \( t \)) holds whenever \( s \) and \( t \) unify (without occurs check). However, in contrast to the behavior of the predicate =/2, the variables in \( s \) and \( t \) should not be instantiated by proving check_unify(\( s \), \( t \)). For example, the query

?- check_unify(X, 1), length([], X).

should succeed with the result \( X = 0 \).

*Hint: You may use the built-in predicate =/2 and the meta-predicate \(+/1\) (or not/1) to solve this exercise.*

Merry Christmas and a Happy New Year!