Remarks:

- For this exercise, you will need SWI-Prolog\(^1\).
- For exercise 1-3, you need to send your solutions as source files to your respective tutor.
- Make sure your source files do properly compile and run!
- Document and explain your code!

Exercise 1 (8 Points, Binomial coefficient)

(a) Write a predicate \texttt{binom/3} that takes two integers \(k, n \in \mathbb{Z}, k \geq 0\) and calculates the binomial coefficient \(\binom{n}{k}\). For example:

\begin{verbatim}
?- binom(3,2,X).
X = 3.
true.
?- binom(5,2,X).
X = 10.
true.
\end{verbatim}

(b) Now assume additionally that \(n \geq 0\) and \(n \geq k\). Use the predicate \texttt{fac/2} from the lecture (Ex. 7.74) to calculate the binomial coefficient. Let your predicate \texttt{binom/3} dynamically switch between both procedures, s.t. the one including \texttt{fac/2} is used whenever possible.

(c) Rework the predicate \texttt{fac/2} so that results are stored in the database (there should never be more than one entry in the database for each value) and are reused whenever possible. For example, suppose we make the following query:

\begin{verbatim}
?- fac(4,X).
X = 24.
true.
?- listing.
...
facresult(4,24).
...
true.
\end{verbatim}

Then, if we go on to ask \texttt{?− fac(5,X)}. Prolog should not calculate everything anew, but should get the result for \texttt{fac(4,X)} from the database and multiply by 5.

Exercise 2 (7 Points, Primes)

(a) A twin prime is a prime number \(p\) such that one of \(p + 2\) or \(p - 2\) is also a prime. Write a predicate \texttt{tprime/1} that is satisfied if the argument is a twin prime.

\(^1\)http://www.swi-prolog.org/
(b) Write a new predicate `primef/2` that takes a nonnegative integer as its first argument and returns its prime factorization as second argument.

(c) Write a predicate `tripleprime/1`, as short as possible, that is satisfied if the argument `n` is a prime, `n + 2` is a prime and `n + 4` is a prime.

Exercise 3 (9 Points, Cut)

Explain for each of the following programs what the effects for the goal `?- f(X).` would be (including any backtracking requested). List the results and explain them.

(a) 
```
f(X) :- !, X = p.  
```
```
f(X) :- !, X = q.  
```
```
f(X) :- X = r.  
```

(b) 
```
f(X) :- X = p, !.  
```
```
f(X) :- X = q, !.  
```
```
f(X) :- X = r.  
```

(c) 
```
f(X) :- X = p.  
```
```
f(X) :- !, X = q.  
```
```
f(X) :- X = r.  
```

(d) 
```
f(X) :- X = p.  
```
```
f(X) :- X = q, !.  
```
```
f(X) :- X = r.  
```

(e) 
```
f(p) :- !.  
```
```
f(q) :- !.  
```
```
f(r).  
```

(f) 
```
f(p).  
```
```
f(q) :- !.  
```
```
f(r).  
```

Exercise 4 (6 Points, SLD and Cut)

Consider the two programs

```
max(X,Y,Y) :- X <= Y, !.  
```
```
max(X,Y,X) :- X > Y.  
```

and

```
max(X,Y,Z) :- X <= Y, !, Y = Z.  
```
```
max(X,Y,X).  
```

both of which determine the maximum of two numbers.

(a) Draw the SLD trees for both programs for the goal `?- max(3, 5, G).` Ignore the cut! (i.e. continue drawing the tree even if the cut would remove a path) Mark in each tree all solutions, that are found, all errors and all unifications.

(b) Mark in each tree where paths are removed by the cut.

(c) What is the difference between both programs? Decide for each cut, whether it is red or green and explain why.