**Team**: Nr. 7, Marjan Bachtiari, Tim Hartig

**Aufgabenaufteilung**:

* Zur Grundlagenübung hat jedes Teammitglied die Aufgaben selbstständig bearbeitet.

**Quellenangaben**: <entfällt>

**Begründung für Codeübernahme**: <entfällt>

**Bearbeitungszeitraum**:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Bearbeitung 1 | Bearbeitung 2 | Bearbeitung 3 |
| Bachtiari |  |  |  |
| Hartig | 28.03. – 2,5 Std. | 29.03. – 2,0 Std. | 30.3. – 3,0 Std. |

**Aktueller Stand**: Implementiert sind Teil 1.1, 1.3 bis 1.5 und 2.1 bis 2.4.

**Skizze**:

Teil 1

1. L ist dann eine Liste, wenn L sich mit einer Liste beliebiger Länge unifizieren lässt.
2. Idee: Über die Listen L1 und L2 nacheinander iterieren (rekursiv) und bei jedem Iterationsschritt das Kopfelement der aktuellen Liste in L3 anhängen.  
   WICHTIG: Das Anhängen muss auf dem „Rückweg“ der Rekursion erfolgen, da sonst das Ergebnis in L3 gleich den umgedrehten Versionen von L1 und L2 entspräche.  
   Wir brauchen eine Regel, mit der sich bis zum Ende einer Liste rekursiv vorgearbeitet wird {Abbruchbedingung: prädikat(\_K|[]). }
3. Infix bedeutet, dass sich das an I gebundene Element innerhalb einer Liste L befinden soll.  
   Da nicht geklärt ist, ob die Sonderfälle Listenlänge kleiner 3, wo es kein echtes „Infix“ geben kann, also ein Dazwischen, werden wir lediglich auf die Existenz des Elements innerhalb der Liste prüfen.
4. Das Element muss vor der, einer Liste abschließenden, leeren Liste vorhanden sein.
5. Das Prädikat muss gelten, wenn das Element I der Kopf der Liste L ist.
6. …
7. Wir gehen rekursiv über die Liste L bis zum Ende (Rest = []) und geben eine „Ergebnis-Liste“ R für die Rückgabe mit.  
   Lässt sich auf dem Rückweg der Rekursion die Regel beweisen, dass sich E mit dem aktuellen Kopfelement von L unifizieren lässt, so wird R ohne Anhängen des aktuellen Kopfelements von L weitergereicht. Andernfalls wird das aktuelle Element an R angehängt.
8. Funktioniert im Prinzip wie 7., nur dass, falls E1 das aktuelle Element von L ist, in R E2 statt E1 angefügt werden soll.

Teil 2

1. Wir nutzen das Prädikat dept und lassen DEPTNO und DNAME variabel und binden LOC an ‘BOSTON‘.
2. Wir brauchen ein Prädikat mit 3 Variablen für DNAME, EMPNO und ENAME. Wir erstellen eine Regel, in der zuerst mit Hilfe von dept() die anhand DNAME die passende Department Nummer an DEPTNO gebunden wird, um damit anschließend alle Mitarbeiter auszugeben, die unter DEPTNO zu finden sind.  
   Dann nutzen wir den Aufruf p(DNAME,EMPNO,ENAME) mit DNAME = ‘RESEARCH‘.
3. Wir nutzen ein Prädikat, das nur eine Variable für den Rückgabewert brauchen wird – für die Namen.  
   Um nach Angestellten mit bestimmten Kriterien zu suchen (hier Job = ‘PRESIDENT‘ <oder> ‘MANAGER‘) müssen wir das Prädikat emp() nutzen und die Variable JOB bei den Aufrufen an die zwei genannten Jobnamen binden, um die Namen zu erhalten. Da wir eine ODER-Verknüpfung zwischen beiden Bedingungen haben, müssen wir den Aufruf mit emp() in zwei Regeln verwenden, in denen je nach einem Job abgefragt wird.
4. Das nötige Prädikat braucht eine Variable für die Ausgabe und wir müssen emp() zwei Mal verwenden. Dabei wird die Ausgabe an den ENAME im ersten emp() gebunden, MGR an die Stelle von EMPNO in der zweiten Verwendung von emp() und es müssen beide Gehälter verglichen werden. Die Regel soll gelten, wenn es einen direkten Vorgesetzten gibt und das Gehalt aus emp() Nr. 1 größer als das Gehalt aus emp() Nr. 2 ist.
5. Das Prädikat braucht eine Variable für den gesuchten MGR und eine für die Ausgabe der Angestellten, die direkt oder indirekt MGR unterstellt sind.  
   Zuerst müssen wir uns, da unter MGR die entsprechende Angestellten-Nummer hinterlegt ist und nicht direkt der Name, die Nummer von ‘JONES‘ holen. Wir binden MGR an ‘JONES‘ und die zugehörige EMPNO an die Stelle MGR in der nächsten Verwendung von emp(), um die unterstellten Mitarbeiter zu finden.  
   …