

Unit Code: CM20019

Unit Lecturer: Dr. P. Bruscoli

Coursework 3 – due 14 December 2006

Please read the following information and instructions for the coursework submission process:

- Date, Time, Location for submission:
This coursework is to be submitted by 14 December, at 5:00 pm both in paper form through the coursework post box located outside 1W2.23 and electronically, per email.
- Form of submission:
Paper Submission: please submit a print out of your Prolog code (and tests) and be sure it contains as first lines (as comments in Prolog) information to identify you: surname, first name, and student number. Please check that the sheets of paper are bound securely. All students must include the coursework signature sheet.
Email Submission: the same text constituting the Paper Submission shall be sent per email to the Unit Lecturer, to the address **P.Bruscoli@Bath.Ac.UK**, with the Subject: **XYZWPL–Student Number**.
Please be aware that submission per email simply constitutes a help for the correction process. Only the paper submission constitutes the official submission.
- The proportion of the total assessment for this coursework:
This is the third of three courseworks, and they all will contribute the 25% of the final grade. This specific coursework contributes 100 points (over a total of 300 points in three courseworks) and contributes for the 8% of the final grade. This includes one exercise composed of three incremental parts, whose points are indicated at their beginning. Points for each part are split, as indicated, between the code and the testing.
- Condition of assessment:
This is an individual coursework, and it can be completed during the laboratories sessions and preferably during the student's own time. Tutors may be consulted according to their meeting times and agreements.
- Specification:
This coursework contains 1 exercise, divided in three incremental parts, that need therefore to be solved in the given order. Each part requires providing a program and running the output produced on some given test cases.
- Deliverables:
Please provide, both on paper and per email, both the code for the programs and the results in running the required tests.
- Marking Guidance:
Questions in this text will be marked proportionally to the points they carry. The sum of all points obtained in this coursework will contribute to the points obtained at the end of all three coursework.
- Feedback:
It will be provided by example solutions, or by discussing solutions during tutorials.

Exercise 1 [points: 20, 20, 20]

Consider the usual definitions of terms and syntactic trees, together with the following notions of ground terms and frontier of a ground term.

Ground terms:

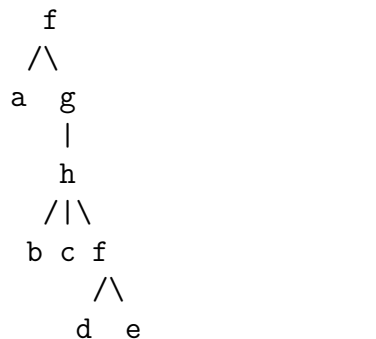
- a is a ground term, where a is a function symbol of arity 0
- $f(t_1, \dots, t_n)$ is a ground term, if f is a function symbol of arity n and t_1, \dots, t_n are ground terms.

Therefore, ground terms are those that do not contain variables.

Given the syntactic tree associated to a ground term, all its leaves return the so-called *frontier* of the tree. We can represent the frontier, more precisely, as the *list* of all the leaves obtained by visiting the tree top-down, in a left-to-right, depth-first strategy.

For example, a ground term is $f(a, g(h(b, c, f(d, e))))$, whose function symbols have the following arities: $f/2, g/1, h/3$ and $a, b, c, d, e/0$.

The syntactic tree associated to this ground term is depicted below, and the frontier of the tree is represented (according to what previously said) as the list `[a, b, c, d, e]`:



Part 1 – [points: 32 (code 20: tests 3, 3, 3, 3)]

1. Write in Prolog a program, to define the predicate

`findFrontier(T,F),`

which, given a ground term T , returns its frontier F .

2. Test your program on the following inputs for T and provide the result you obtain:
Test 1: `a`

Test 2: `f(a,f(b,a))`
Test 3: `f(g(a), g(f(b,g(c))))`
Test 4: `f(g(h(b,c,f(h(d,l,g(m))),a,e)))`

Part 2 – [points: 36 (code 20; tests 2, 2, 2, 2, 2, 2, 2, 2)]

1. Write in Prolog a program, to define the predicate

`oneOrdered(L),`

which, given a list of 0-ary function symbols (constants), succeeds iff each component of the list `L` is one single letter, and all components are alphabetically ordered.

(So, for example, the query `?-oneOrdered[a,e,g,m].` will succeed, whereas queries `?-oneOrdered[e,a,g,m].` and `?-oneOrdered[i,need,help].` shall fail.)

2. Test your program on the following inputs for `L` and provide the result you obtain:
Test 1: `[]`
Test 2: `[A]`
Test 3: `[a]`
Test 4: `[abc, def, ghi, jkl, mno]`
Test 5: `[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,z,y]`
(be careful to the order of the last two)
Test 6: `[a,B,c,d,E]`
Test 7: `[a,a,a,b,b,r,s,t,t,v]`
Test 8: `[aB,zY]`

Part 3 – [points: 42 (code 20; tests 6, 6)]

1. Write in Prolog a program, to define the predicate

`makeBinTree(T1,T2),`

with the following specification: it transforms a ground term `T1`, whose frontier `F1` satisfies `oneOrdered(F1)`, into a new ground term `T2`, standing for a binary tree generated from `T1`, such that

- the frontier `F2` of `T2` is obtained by the frontier `F1` of `T1` by deleting all duplicates;

- if F2 contains just one 0-ary function symbol, then T2 will just be that symbol;
- if F2 contains more than one 0-ary function symbols, then T2 will use just one binary function symbol $t/2$ as term constructor.

(For example the query `?-makeBinTree(f(a,g(h(a,a,f(a,a))))),T2)` will return `a`, whereas the query `?-makeBinTree(f(a,g(h(b,c,f(d,e))))),T2)` can return `t(a,t(b,t(c,t(d,e))))` (depending on how you build the binary tree).

2. Test your program on the following inputs for T1 and provide the result you obtain:

Test 1: `f(a,g(h(a,a,f(a,a))))`

Test 2: `f(a,g(h(b,c,f(d,e))))`