BROCK UNIVERSITY

Final exam
Course: COSC 2P93 Logic Programming
Date of exam: Monday, April 26, 2003
Time of exam: 1900-2200

pages: 13
students: 55
hours: 3
Instructor: B. Ross

NAME (print): ___________________________________

STUDENT NUMBER: ________________________________

There are 6 questions totalling 100 marks.
No aids are permitted. Use or possession of unauthorized materials will automatically result in a grade of zero for this examination.
Please answer all questions on the exam paper. Use the backs of pages if necessary.
Read the questions carefully. Keep written answers brief and to the point. Write legibly.
For programming questions, feel free to write as many auxiliary predicates as necessary.
You may assume that the following builtin predicates are available: sort/2, length/2.
Do not assume any other predicates are available (append, member, etc.)
A grade of 40% is required on this exam to pass the course.

Question	**Total**	**Mark**
1	10	
2	16	
3	16	
4	18	
5	16	
6	24	
TOTAL:	**100**	

Question 1 [10] Define the following terms as discussed in the course:

(a) program database

(b) logical variable

(c) singleton variable

(d) unification

(e) failure-driven loop

Question 2 [16] Consider the following buggy program:

```
makeDups(0, C, C).
makeDups(N, C, [C|R]) :-
        N > 0,
        N is N-1,
        makeDups(N, C, R).
```

The intension is that, in the query “?- makeDups(N, C, L)”, a total of N copies of element C are placed into list L. For example, “?- makeDups(5, cat, L)” should result in L=[cat,cat,cat,cat,cat].

(a) [4] What actually occurs with the query “?- makeDups(5, cat, L)”? What part of the code is causing this to happen?

(b) [4] Correct the program so that it behaves as expected. Make the least number of corrections possible (in other words, please don’t rewrite the entire predicate from scratch!)

(c) [4] Modify your corrected version of the predicate so that it fails when either N is the value 6, or C is the constant ‘dog’. Backtracking should not process these values in any way.

(d) [4] Is your program in (b) determinate or nondeterminate? Discuss.

Question 3 [16]

(a) [6] Write a predicate factorial/2. It computes the factorial of an integer N > 0. For example, “?- factorial(4, F)” computes F=24 (because 4! = 4*3*2*1 = 24).

(b) [10] Write a predicate bigFactorial/2. Given a list, it computes the factorial for every positive integer in the list, returning the factorial in another list. Non-positive integers are skipped. For example,

```
?- bigFactorial([1,2, dog, 3,-10, 4], L).
L = [1, 2, 6, 24]
```

Question 4 [18] Consider this predicate for removing duplicates from lists:

```
remove_dups([], []).
remove_dups([A|R], S) :-
        member(A, R),
        remove_dups(R, S).
remove_dups([A|R], [A|S]) :-
```

(a) [4] Given the query "?- remove_dups([1,2,2,3], L).", the first solution is L=[1,2,3]. However, on backtracking, the next and final solution is L=[1,2,2,3]. Explain precisely why this occurs in the above program.

(b) [4] Using cut(s), modify the program to remove the above undesirable backtracking behavior. If you modify the code above, clearly indicate the changes.

Question 4 (cont)
(c) [4] Rewrite the program to remove the same undesirable backtracking behavior, but this time, make the program strictly declarative (pure Prolog!).

(d) [6] Write a predicate test_dups/2. Given a list, it counts the number of duplicates in the list, and returns that count as the result. For example, “?- test_dups([1,2,2,3,4,4,5], A)” computes A=2 (ie. there’s an extra 2 and 4). This predicate can be declarative or procedural.

Question 5 [16] Write a program histogram/2, which takes a file name, and a list of integers. All the output is written to the given file. The program first writes a graphical histogram, where each row corresponds to each number in the list (see below). Then histogram finds the average of the list, and writes this row out as well, with a blank line between it and the prior rows. For the query, “?- histogram(myfile, [1,2,2,3,4,2,5,1]).”, the file ‘myfile’ will have the following written to it:

```
* 1
** 2
** 2
*** 3
**** 4
** 2
***** 5
* 1

** avg=2
```

Question 5 (cont)

Question 6 [24] This question involves writing a program to solve the famous Knapsack problem, which is a classic combinatorial problem in computer science. Read the entire question before starting it!

While roaming around on a beach in the Caribbean, you find a treasure chest. Each relic in the treasure chest has a weight (kg) and value (US $). You have a knapsack that can hold a maximum of 100 kg. You must fill the knapsack with as many items as possibly, such that the 100 kg weight limit is not exceeded. Naturally, it is better for you if you select the items that have the highest value. For example, a 1 kg gold plate worth $1000 is worth choosing over a 1kg pail of rancid pudding worth only $1.

(a) [16] Write a predicate knapsack/3. It works as follows. There exists a fact that lists all the items available in the treasure chest. For example,

```
treasureChest([item(5, 25), item(50, 100), item(40, 50), item(60, 90),
                        item(20, 100), ...]).
```

For item(5,25), its weight is 5kg, and its value is $25. Then, knapsack/3 finds legal selections of items which pack the knapsack as fully as possible without exceeding the 100kg total weight. When it returns a solution, it means that no item is left in the chest that can fit with the given items, without exceeding 100kg. Backtracking to knapsack/3 returns other combinations of items. Furthermore, it will not return permutations of the same set of items. Every returned result is unique. Please note that this predicate does not try to find the best set of items (see b below!).

For example:

```
?- knapsack(Knapsack, Weight, Value).
Knapsack = [item(5, 25), item(50, 100), item(40, 50)], Weight = 95, Value = 175 ;
Knapsack = [item(5, 25), item(60, 90), item(20, 100)], Weight = 85, Value = 215 ;
etc.
```

Knapsack is a list with all the booty, and Weight and Value are the sum of the weight and value of this knapsack. With the treasure chest shown above, you should not return the solution Knapsack = [item(5, 25)], because there are still available items in the chest that will fit in this nearly-empty knapsack. For example, item(50,100) will fit with item(5, 25), as will item(40,50). But once you pack these 3 items, the weight is 95kg, and you can't fit anything more in (at least with respect to the list of items in the treasure chest shown).

(b) [8] Using your predicate in (a), write a predicate that finds the best selection of booty! This is the knapsack that has the highest value, while not exceeding the 100kg weight limit.

Question 6 (cont)

Question 6 (cont)

Question 6 (cont)

***** student_mood(happy) :- grade(2p93, ‘A’), term(over), !. *****