UNIVERSITY OF PUERTO RICO AT BAYAMON

DEPARTMENT OF COMPUTER SCIENCE

**COTI 4039 – COMPARATIVE PROGRAMMING LANGUAGES**

**ASSIGNMENT #6 – 80 points**

**NOTE**: It is not allowed to use features of the programming language that have not been discussed in class, assigned to read or explicitly authorized.

**I. (20 points) Basic Logic Programming in Prolog**

Using the supplied knowledge base of ***fictitious*** travel information (travel.pl):

1. (5 points) Write a predicate travelByCar/2 which recursively determines whether it is possible to travel by car between two places. The following is an example query and its answer:

?-***travelByCar(bayamon, carolina).***

true

1. (5 points) Write a predicate travelByTrain/2 which recursively determines whether it is possible to travel by train between two places. The following is an example query and its answer:

?-***travelByTrain(bayamon, cabo\_rojo).***

true

1. (5 points) Write a predicate travelByPlane/2 which recursively determines whether it is possible to travel by plane between two places. The following is an example query and its answer:

?-***travelByPlane(carolina, tokio).***

true

1. (5 points) Write a predicate travel/2 which determines whether it is possible to travel from one place to another by chaining together car, train, and plane journeys. The following is an example query and its answer:

?-***travel(bayamon, new\_york).***

true

**II. (40 points) Arithmetic and List Processing in Prolog**

Create a knowledge base called arith\_lists.pl that contains the following:

1. (10 points) A predicate fibo/2 whose last argument is the *n*-th Fibonacci term. Remember that *fibo0 = 0*, *fibo1 = 1*, and *fibon = fibon-1 + fibon – 2*, for *n* > 1. The following is an example query and its answer:

?-***fibo(7, Fibo).***

Fibo = 13

You are required to write two versions: one that uses regular recursion (do not try to find terms much larger than fibo30 or you may hang the machine!) and one that uses a tail-recursive helper predicate.

1. (10 points) A predicate scalar\_mult/3 whose last argument is the result of multiplying an integer by each member of a list of integers representing a vector. The following is an example query and its answer:

?-***scalar\_mult(3, [2,7,4], Mult).***

Mult = [6,21,12]

You are required to write two versions: one that uses regular recursion and one that uses a tail-recursive helper predicate.

1. (10 points) A predicate list\_min/2 whose last argument is the minimum element in a list of numbers. The following is an example query and its answer:

?-***list\_min([5,10,4,3,2,9,1,6,-7,8], Min).***

Min = -7

You are required to write two versions: one that uses regular recursion and one that uses a tail-recursive helper predicate. **Hint:** There is a built-in min/2 function that you could use.

1. (5 points) A predicate list\_insert/3 whose last argument is the result of inserting the given element in the given sorted list of numbers. The following is an example query and its answer:

?-***list\_insert (4, [1,2,3,5], Inserted).***

Inserted = [1,2,3,4,5]

1. (5 points) A predicate insertion\_sort/2 whose last argument is the sorted equivalent of the given list of numbers. It sorts the list by calling list\_insert/3 to insert each element in its corresponding place. The following is an example query and its answer:

?-***insertion\_sort([5,10,4,3,2,9,1,6,-7,8], Sorted).***

Sorted = [-7,1,2,3,4,5,6,8,9,10]

**III. (20 points) Binary Search Tree Processing in Prolog**

Modify binary\_tree.pl to add the following:

1. (5 points) A predicate construct\_tree/2 whose last argument is the binary search tree that corresponds to the given list. The following is an example query and its answer:

?-***construct\_tree([3,4,2,1], Tree).***

Result = node(3,node(2,node(1,nil,nil),nil),node(4,nil,nil))

1. (5 points) A predicate size\_tree/2 whose last argument is the number of elements in the given binary search tree. **Hint:** There is a built-in min/2 function that you could use. The following is an example query and its answer:

?-***size\_tree(Tree, Size).***

Size = 4

1. (5 points) A predicate depth\_tree/2 whose last argument is the number of levels in the given binary search tree. The following is an example query and its answer:

?-***depth\_tree(Tree, Depth).***

Depth = 3

1. (5 points) A predicate min\_tree/2 whose last argument is the minimum element in the given binary search tree. It should use a tail-recursive helper predicate. The following is an example query and its answer:

?-***min\_tree(Tree, Min).***

Min = 1