CS 386 Artificial Intelligence Lab
04/08/2015, Tuesday
Lab Handout 3

General Instructions

- Create a folder named **YOURROLLNUM_LAB3** (e.g. 131050001_Lab3) in the home directory (or at a location of your choice). You need to submit a compressed (tar.gz or tgz only) copy of this folder at the end of this lab.
- This lab requires you to write prolog predicates for **three** problems. In addition, every problem also has an additional part for bonus marks.
- The deadline for submission of the required part (or whatever you are able to complete) is end of today's lab (**Tuesday, 04/08/2015, 5:30 PM**). Solution for the additional/bonus part can be submitted till (**Sunday, 09/08/2015, 11:55 PM**). Separate links will be provided for both submissions. Your final submission should contain all the required (six) predicates.
- More specific submission instructions for each part are provided later.
- All problems must be solved using Prolog only.
- Use of in-built library functions is **not** permitted.
- Strictly follow the predicate names and conventions as described in the problem description and examples.
- Your submission will be automatically graded. Make sure that you read and understand the submission instructions before making a final submission.
- Add comments to your code. Write your Name/Roll number and explain your code briefly.
- A *prolog* source file (say test1.pl) can be compiled and loaded using the list abbreviation.

  ```
  ?- [test1].
  ```

  Do not forget the period at the end.
- You are encouraged to post your queries/doubts about the handout (or anything else) on the **moodle discussion forum** during the lab as well.

Problem 1: Coin Change Problem.

Write a prolog script to output all the ways in which a given amount of money can be made with a list of coins of given denomination . Each denomination is unique, positive and can be used as many times as possible. The list is in sorted order.

The name of the script should be *yourrollnumber_problem1.pl*

The predicate should look like this: *cc(Amount , List , Ans).*

Hints:- *Use relational operators, such as > ,< ,>= or =<.*

For example:-

```
?- cc(20, [5, 9, 15, 11, 20], Ans ).
Ans = [0, 0, 0, 0, 1] ;
Ans = [0, 1, 0, 1, 0] ;
Ans = [1, 0, 1, 0, 0] ;
Ans = [4, 0, 0, 0, 0] ;
false.
```

The numbers in the list shows the number of times, corresponding denomination is used.

Additional Part

Count the total total **number of ways** in which a given amount of money can can be made with a list of coins of given denomination . In this problem, you have to output the total possible actual combinations. Also, this question makes the same assumptions.

Add the new predicate to the same script (*yourrollnumber_problem1.pl*)

The predicate signature should be *nways(Amount , List , Ans).*

For example:-

```
?- nways(20,[5,9,11,15,20],Ans).
Ans = 4 ;
```

<u>Problem 2</u>: Subset Problem

Given a multiset(can contain duplicate elements) and a sum, find the subset of the set whose elements add up to that sum. Each occurrence of an element can be used only once. Elements of the multiset can be assumed to be positive. You can assume that all the elements of the multiset are positive.

Write the predicates in ayour predicate in a script named *yourrollnumber_problem2.pl*

<u>Hints</u>:- *Use relational operators, such as > ,< ,>= or =<.*

A sample query would look like:

```
?- sSet([1,2,5,7],7,Ans).
Ans = [7] ;
Ans = [2, 5] ;
false.
```

<u>Additional Part</u>

Modify the above predicate to can handle multisets that contain negative elements.as well. Add the new predicate to the same file as mentioned above (*yourrollnumber_problem2.pl*).

For Example:

```
?- sSet([1,2,5,7,-7,6,-7],7,Ans).
Ans = [7] ;
Ans = [2, 5] ;
Ans = [2, 5, 7, -7] ;
Ans = [2, 5, 7, -7] ;
Ans = [1, 6] ;
Ans = [1, 7, 6, -7] ;
Ans = [1, 7, -7, 6] ;
Ans = [1, 2, 5, 6, -7] ;
Ans = [1, 2, 5, -7, 6] ;
Ans = [1, 2, 5, 7, -7, 6, -7] ;
false.
```

Problem 3: Legal Knight Moves.

Position of a piece on a chessboard can be represented using a pair *[X, Y]* with $1 \leq X \leq 8$, $1 \leq Y \leq 8$. For example, The lower left corner is represented by [1, 1] while the upper right corner is represented with [8, 8].

Write a predicate (in a file named *yourrollnumber_problem3.pl)* to output all legal moves of a knight from a given position.

Hints:- *Use relational operators, such as > ,< ,>= or =<.*

Example:

```
?- kmove([1,1], Ans).
Ans = [2, 3];
Ans = [3, 2];
false.

?- kmove([3,2], Ans).
Ans = [4, 4];
Ans = [5, 3];
Ans = [5, 1];
Ans = [1, 1];
Ans = [1, 3];
Ans = [2, 4];
```

```
false.
```

Additional Part

Write a similar predicate ***bmove([X,Y], [A,B])*** to output all legal moves of a **bishop** from a given position. Append the predicate to *yourrollnumber_problem3.pl*

Submission

At this point the folder yourrollnum_lab3 should contain three files:

1. yourrollnum_problem1.pl containing your solution to the coin change problem.
2. yourrollnum_problem2.pl containing your solution to the subset sum problem.
3. yourrollnum_problem3.pl containing your solution to the legal knight moves problem.

Compress the folder using the following command

```
tar -zcvf yourrollnum_lab3.tgz yourrollnum_lab3
```

Upload the generated .tgz file as your submission for this lab.

If you complete and submit all six predicates during today's lab, no separate submission is required on Sunday. However, your final submission should contain all the predicates.