Prova d'esame del 17/02/2017

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```
q(Y,X) :- t(X,_), h(Y), not r(Y,X).
r(X,Z) :- s(X,Z), not q(X,Z).
t(X,Y)|h(X):-q(Y,X).

s(1,2). h(1). t(2,2).
```

b) Si aggiungano i seguenti weak constraint, e si calcoli quindi il costo di ogni answer set del programma risultante. Indicare quindi quello ottimo (o quelli ottimi, se più di uno).

```
% DLV syntax                          % ASP-Core-2 syntax
:~ h(X), #sum{Y:q(X,Y)}>1. [X:1]      :~ h(X), #sum{Y:q(X,Y)}>1. [X@1]
:~ t(X,Y), r(_,X). [1:X]              :~ t(X,Y), r(_,X). [1@X,X,Y]
```

c) Si aggiunga ora il seguente strong constraint. Illustrare come cambiano gli answer set, e calcolare nuovamente quello ottimo (o quelli ottimi, se più di uno).

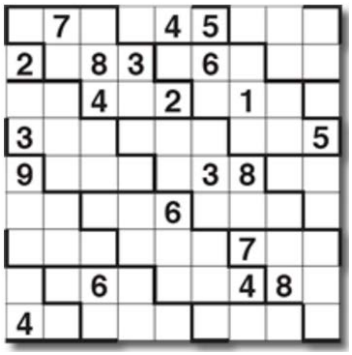
```
:-#sum{Y,X:t(X,Y)}<2.
```

Esercizio 2.
Scrivere un programma ASP che, dato in input un grafo orientato aciclico **G=<N,A>**, etichetti i nodi in **N** con valori compresi tra **1** e **|N|**, in modo tale da soddisfare le seguenti condizioni: 1) indicando con **v(x)** il valore assegnato ad un nodo **x**, per ogni arco **(a,b)** in **A**, si deve avere che **v(b)>v(a)**; 2) indicando con **D(a,b)** la differenza **v(b)-v(a)** per ogni arco **(a,b),** il massimo tra i valori **D(a,b)** deve essere il più piccolo possibile.

Modello dei dati in input:
node(X) ← i nodi del grafo
arc(X,Y) ← gli archi

Prova d'esame del 17/02/2017

Esercizio 3. Scrivere un programma ASP che consenta di risolvere uno schema di Sudoku toroidale, una variante del Sudoku classico, in cui la griglia in input non è divisa in riquadri ma in aree di forma irregolare. Lo scopo del gioco è riempire la griglia di dimensione N*N con numeri da 1 a N, così che ogni riga, colonna o area contenga tutti i numeri da 1 a N senza ripetizioni. Si noti che alcune aree possono essere "aperte" (si veda figura). Le aree aperte in alto proseguono in basso (e viceversa) nella stessa colonna, le aree aperte a destra proseguono a sinistra (e viceversa) nella stessa riga. **ESEMPIO:** la figura mostra uno schema di sudoku toroidale 9*9 (a sinistra) e la sua soluzione (a destra).



Prova d'esame del 03/04/2017

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```
pollo(X) | limone(X) :- finocchio(X), not mela(X).
mela(X) :- finocchio(X), not pollo(X).
finocchio(X) :- pistacchio(X,Y), mela(Y).

mela(1). finocchio(3). pistacchio(2,3). pistacchio(4,2). pollo(4).
```

b) Si aggiunga il seguente weak constraint, e si calcoli quindi il costo di ogni answer set del programma risultante. Indicare quindi quello ottimo (o quelli ottimi, se più di uno).

```
% DLV syntax                          % ASP-Core-2 syntax
:~ pollo(X), finocchio(X). [1:X]      :~ pollo(X), finocchio(X). [1@X]
```

c) Si aggiunga ancora il seguente weak constraint:

```
% DLV syntax                          % ASP-Core-2 syntax
:~ mela(X), not limone(X). [2:X]      :~mela(X), not limone(X). [2@X]
```

Calcolare nuovamente gli answer set riportando per ciascuno il costo e indicare quello ottimo (o quelli ottimi, se più di uno).

d) Si aggiunga ancora il seguente strong constraint.

```
:- pollo(Y), #sum{X: pistacchio(Y,X), finocchio(X)}<2.
```

Come influisce sulle soluzione del programma? Perché? Motivare adeguatamente la risposta.

Esercizio 2.
Scrivere un programma ASP che, dati in input:

- un grafo orientato aciclico **G=<V,E>**;
- una funzione di costo che assegni un valore intero positivo a ciascun arco in E;
- due sottoinsiemi *S* e *D* di V, rispettivamente indicati come sorgenti e destinazioni, tali che a ciascun elemento di S corrisponda un elemento in D, e viceversa (quindi sono individuate coppie sorgente-destinazione);

individui un sottoinsieme E' degli archi in E tale che ogni coppia sorgente-destinazione risulti CONNESSA in E' (in altre parole, per ogni coppia di nodi sorgente-destinazione <s,d> deve esistere un cammino da *s* a *d* composto da archi contenuti in E'). Si vuole inoltre minimizzare il peso totale del sottoinsieme (cioè la somma dei pesi degli archi in E').

Prova d'esame del 03/04/2017

Modello dei dati in input:

node(X)	←	i nodi del grafo
arc(X,Y,W)	←	gli archi con i rispettivi pesi
source(S,Pos)	←	il nodo "S" è la sorgente numero "Pos"
destination(D,Pos)	←	il nodo "D" è la destinazione numero "Pos" (una sorgente ed una destinazione sono accoppiati se hanno la stessa posizione)

Esercizio 3. Scrivere un programma ASP che consenta di risolvere uno schema del rompicapo descritto di seguito. Viene data una griglia di dimensione NxN in cui posizionare delle navi secondo il classico schema del gioco della "battaglia navale". In particolare, ci sono 10 navi da disporre: una di dimensione 4, due di dimensione 3, tre di dimensione 2, quattro di dimensione 1. Per "dimensione" di una nave si intende il numero di celle che questa occupa. Si noti che, nel disporre le navi, bisogna assicurarsi che nessuna nave si sovrapponga ad un'altra (su una cella può essere posizionato al massimo il pezzo di una nave, non più di uno). Inoltre, in alcune delle celle della griglia sono presenti dei numeri. Le navi devono essere posizionate in modo tale che, se in una cella è presente il numero "k", allora "k" celle adiacenti ad essa sono occupate da navi.

Prova d'esame del 23/06/2017

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```
andiamo(X) | a(X) :- comandare(_,X), X < 4.
comandare(U,Z)  :- b(U), not trattore(U), b(Z).
trattore(W) :- b(X), not comandare(W,X), W = X - 1.
b(3).
b(4).

:- a(X), #count{Y,Z : comandare(Y,Z)} > X.
```

b) Si immagini ora di dover preferire gli answer set in cui non "andiamo sul trattore": cioè preferiamo le soluzioni in cui non sono contemporaneamente veri "andiamo" e "trattore" per lo stesso "X". Si modifichi il programma adeguatamente aggiungendo uno o più weak constraints e si calcolino i costi di ciascun answer set, indicando quello ottimo (o quelli ottimi).

c) Si immagini ora che ci si accorga che è *ancora più importante* "comandare" solo se "andiamo": cioè, non ci piacciono le soluzioni in cui "comandare" è vero per qualche "X" al secondo parametro, ma "andiamo" è falso per lo stesso "X".

Esercizio 2.
Siano dati in input:

- un grafo *non orientato* **G=<V,E>**;
- un insieme di colori.

Si scriva un programma ASP che calcoli il numero acromatico di G. Il "numero acromatico" di un grafo G è il massimo numero di colori differenti con i quali è possibile colorare i nodi di G rispettando le seguenti direttive:

- ciascun nodo deve essere colorato con esattamente un colore;
- per ogni coppia di nodi agli estremi di un arco si devono scegliere due colori distinti;
- per ogni coppia di colori distinti C1 e C2, tra quelli usati nella colorazione, deve esistere almeno un arco in E i cui due vertici sono colorati con C1 e C2, rispettivamente.

Si ricordi che tra tutte le colorazioni possibili quella di interesse è quella in cui il numero di colori usati è massimo.

Modello dei dati in INPUT:

- node(X) ← i nodi in V del grafo in input
- arc(X,Y) ← gli archi in E del grafo in input
- color(X) ← i colori a disposizione per costruire le colorazioni

Modello dei dati in OUTPUT:

- achromaticNumber(X) ← il numero acromatico del grafo in input

Prova d'esame del 14/07/2017

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```
granite(X) | granate(X,Y) :- lasciate(X), speranze(X,Y).
speranze(X,Y) :- entrate(X,Y), not e(X), state(Y).
e(X) :- speranze(_,X), not granite(X).
granite(X) :- not e(X), state(X).

state(2). state(3).
lasciate(3).
entrate(1,2). entrate(1,3).
```

b) Si immagini ora che si debba vietare di avere più di una granita. Si modifichi adeguatamente il programma in modo che abbia answer set conformi a quanto specificato.

c) Si aggiungano ora i seguenti weak constraint al programma ottenuto al punto (b):

```
:~ #sum { Y,X : speranze(X,Y), e(Y) } = N, entrate(Z,N). [ N : 1 ] %/ [N@1, Z,N]
:~ e(X). [ 1 : X ] %/ [1@X, X]
```

Si calcolino i costi per tutti gli anwer set, indicando quello ottimo e commentando il procedimento seguito.

Esercizio 2.
Sia dato un grafo orientato **G=<V,E>** tale che gli archi in *E* siano pesati (pesi solo positivi), e che l'insieme dei nodi *V* sia tri-partito in tre sottoinsiemi *A*, *B* e *C*; i sottoinsiemi sono tali che tutti gli archi uscenti dai nodi in *A* incidono solo su nodi in *B*, e tutti gli archi uscenti dai nodi in *B* incidono solo su nodi in *C*. Il grafo è tale che da ogni nodo in A si può raggiungere qualunque nodo in C. Rimuovendo una parte dei nodi in B (e quindi anche gli archi entranti/uscenti corrispondenti), si intende trovare un sotto-grafo di *G* che abbia le seguenti caratteristiche:

- (condizione necessaria): la completa raggiungibilità da A a C è mantenuta;
- (cosa più importante): la cardinalità di B è minima;
- (cosa meno importante): massimizzare il peso degli archi che da B vanno in C.

Modello dei dati in INPUT:

- node(X,I) ← i nodi in V del grafo in input, dove I indica il gruppo (a,b o c)
- arc(X,Y,W) ← gli archi in E del grafo in input, dove W indica il peso

Modello dei dati in OUTPUT:

- keepNode(X,I) ← i nodi da tenere
- keepArc(X,Y,W) ← gli archi da tenere
- cardinalityOfB(X) ← il numero di nodi in B nel grafo ottenuto
- totalGainBtoC(X) ← la somma totale dei pesi degli archi da B in C.

87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410

Prova d'esame del 08/09/2017

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```
a(L,L) | b(L,L) :- c(L), L>1.
a(L,M) :- b(M,L), c(L), L<3.
b(M,L) :- a(M,X), c(L), not c(X).
c(2).   c(3).   b(1,2).   b(2,3).
```

b) Si aggiunga ora il seguente weak constraint al programma indicato al punto (a), si calcolino i costi per tutti gli answer set, indicando quello ottimo e commentando il procedimento seguito.

```
:~ a(X,Y). [ X : Y ]
```

c) Si immagini ora di voler minimizzare (al livello di priorità più basso) i casi in cui compaiono coppie del tipo "b(W,Z), a(Z,W)". Si definisca un opportuno weak constraint, lo si aggiunga al programma ottenuto al punto (b) e si calcolino nuovamente i costi per tutti gli answer set, indicando quello ottimo e commentando il procedimento seguito.

Esercizio 2.
Si scriva un programma ASP che risolva il problema seguente: dato un insieme di persone V, una relazione (simmetrica) di amicizia tra essi ed un insieme di tavoli T, ciascuno avente esattamente 3 posti, assegnare alcuni delle persone di V ai tavoli, tenendo conto di quanto esposto di seguito.

- Non tutte le persone devono per forza finire seduti ad un qualche tavolo.
- Ciascun tavolo deve essere lasciato vuoto oppure occupato interamente da 3 persone.
- Se 3 persone sono sedute allo stesso tavolo, tutte sono amiche tra loro (cioè, se ***a***,***b***,***c*** sono allo stesso tavolo, allora ***a*** deve essere amico di ***b*** e di ***c***; ***b*** di ***c*** di ***a***, e ***c*** di ***a*** e di ***b***).
- Si vuole massimizzare il numero di tavoli usati (cioè non vuoti).

Modello dei dati in INPUT:

- persona(X) ← le persone da far sedere
- amici(X,Y) ← la relazione di amicizia tra due persone X e Y
- tavolo(T) ← i tavoli a disposizione

Prova d'esame del 20/11/2017

Esercizio 1. Svolgere tutti i punti.

a) Si consideri il seguente programma logico e se ne calcolino gli answer set, illustrando adeguatamente il procedimento seguito.

```
r(Z) :- s(X), Z=X+1, not p(Z).
s(X) :- r(X), not p(X).
p(X) | t(X) :- q(Y,X), not s(Y).

r(2). p(4). q(1,2). q(1,3).
```

b) Si aggiunga ora il seguente constraint al programma indicato al punto (a) e si calcolino gli answer set del programma ottenuto, illustrando adeguatamente il procedimento seguito.

```
:-  #sum{ X,Y:s(X),p(Y)}>=5.
```

c) Si aggiunga ora il seguente weak constraint (riportato con due sintassi alternative) al programma ottenuto al punto (b) e si calcolino i costi per tutti gli answer set, indicando quello ottimo e commentando il procedimento seguito.

```
:~ p(X), s(Y). [Y : X]                    :~ p(X), s(Y). [Y@X, X,Y]
```

Esercizio 2.
Si scriva un programma ASP che risolva il problema seguente: dato un grafo (V,A) e un insieme di coppie di nodi C, calcolare un sottoinsieme P dei nodi V tale che tutte le seguenti condizioni siano rispettate.

- Ogni coppia di nodi di P deve essere connessa, ossia deve esistere un cammino tra i due in A.
- Per ogni coppia di nodi (C1,C2) in C, al più uno tra C1 e C2 deve appartenere a P.
- La cardinalità di P deve essere la più grande possibile.

È richiesto inoltre di produrre in output un predicato che conservi la cardinalità di P.

Modello dei dati in INPUT:
node(X) ← i nodi in V
arc(X,Y) ← gli archi in A
pair(X,Y) ← le coppie in C

Modello dei dati in OUTPUT:
in(X)← i nodi in P
length(L)←la cardinalità di P.

87036 Arcavacata di Rende (CS) - Ponte Pietro Bucci Cubo 30B
tel. (+39) 0984 496402 – fax (+39) 0984 496410