Esercizio 1. Svolgere tutti i punti.
a-1) Si consideri il seguente programma logico e se ne calcolino tutti gli answer set, ***illustrando adeguatamente il procedimento seguito***.

```
t(X,Y)  :- s(X,Y,Z), not q(X,Y).    1
q(X,Y)  :- not t(X,Y), s(X,Y,Z).    2
s(Z,X,X):-q(X,X),t(X,Z).   3
s(1,1,1).
s(1,2,3).
```

a-2) Si aggiunga il seguente strong constraint al programma del punto precedente.

```
:-#max{X,Y:s(X,Y,Z)}>1.
```

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

t(1,1) :- ~~s(1,1,1)~~, not q(1,1).
t(1,2) :- ~~s(1,2,3)~~, not q(1,2).
q(1,1) :- not t(1,1), ~~s(1,1,1)~~.
q(1,2) :- not t(1,2), ~~s(1,2,3)~~.
~~s(1,1,1) :- q(1,1), t(1,1).~~ % é un fatto
s(2,1,1) :- q(1,1), t(1,2).

t(2,1) :- s(2,1,1), not q(2,1).
q(2,1) :- not t(2,1), s(2,1,1).

~~s(1,1,1) :- q(1,1), t(2,1)~~

* il grounding termine

* vedi semplificaz. ↑

AS: {
A1: { s(1,1,1), s(1,2,3), t(1,1), t(1,2) },
A2: { s(1,1,1), s(1,2,3), t(1,1), q(1,2) },
A3: { s(1,1,1), s(1,2,3), q(1,1), q(1,2) },

A4: {s(1,1,1), s(1,2,3), t(1,2), q(1,1), s(2,1,1), t(2,1)}

A5: {s(1,1,1), s(1,2,3), t(1,2), q(1,1), s(2,1,1), q(2,1)}

}

a-2)

:- #max {X,Y : s(X,Y,Z)} > 1

A1 : 1 > 1.
A2 : 1 > 1.
A3 : 1 > 1.
A4 : 2 > 1. ✗
A5 : 2 > 1. ✗

Inserendo lo strong constraint verranno scartati A4 ed A5

b) Si consideri ora un programma P (non è necessario sapere come è fatto) i cui answer set sono già stati calcolati e sono riportati di seguito.

```
A1: {g(1), g(2), f(1,1), c(1,2), f(1,2), c(2,2)}

A2: {g(1), g(2), f(1,1), f(2,1), f(1,2), f(2,2)}

A3: {g(1), g(2), f(1,1), c(1,2), f(1,2), f(2,2)}
```

Si supponga di aggiungere i seguenti weak constraint al programma P. Si calcoli quale sarebbe il costo di ognuno degli answer set riportati sopra: *si riporti il* ***costo dettagliato per ciascun answer set*** *e si indichi quello* ***ottimo***, ***commentando adeguatamente il procedimento seguito***.

```
:~ f(X,Y). [X@Y,X,Y]
:~ c(X,Y),g(Y). [Y@X,X,Y]
```

A1: 3@1 3@2
:~ f(1,1). [1@1, 1,1]
:~ f(1,2). [1@2,1,2]
:~ c(1,2), g(2). [2@1, 1,2]
:~ c(2,2), g(2). [2@2, 2,2]

A2: 3@1 3@2
:~ f(1,1). [1@1, 1,1]

```
:~ f(1,2).    [1@2,1,2]
:~ f(2,1).    [2@1,2,1]
:~ f(2,2).    [2@2,2,2]
```

A3: 3@1 3@2

```
:~ f(1,1).    [1@1,1,1]
:~ f(1,2).    [1@2,1,2]
:~ f(2,2).    [2@2,2,2]
:~ c(1,2), g(2).  [2@1,1,2]
```

In questo caso, tutti e tre sono considerati AS ottimi, in quanto a parità di livello di priorità, pagano in egual modo.

Esercizio 2. Sia **G** = **<V, E>** un grafo orientato, dove **V** è l'insieme dei nodi ed **E** l'insieme degli archi. Identificare, *scegliendo accuratamente tra gli archi di* ***E***, un sottografo di **G** detto **G1** = **<V, E1>** avente le seguenti proprietà:

1) per ogni coppia di vertici **u** e **v** in **V**, nel grafo **G1** esiste un cammino da **u** a **v** se e solo se anche in **G** esiste un cammino da **u** a **v**.
2) La cardinalità di **E1** è minima.

MODELLO DEI DATI IN INPUT

nodo(X)	← i nodi in **V**
arco(X,Y)	← gli archi in **E**

```
in(X,Y) | out(X,Y) :- arco(X,Y).

:- #count{X,Y : in(X,Y)} = 0

reaches(X,Y) :- nodo(X), nodo(Y), arco(X,Y).
reaches(X,Z) :- nodo(X), nodo(Y), nodo(Z), reaches(X,Y),
                arco(Y,Z).

reaches_g1(X,Y) :- nodo(X), nodo(Y), in(X,Y).
reaches_g1(X,Z) :- nodo(X), nodo(Y), nodo(Z), reaches_g1(X,Y)
```

```
reaches_g1(X,Z) :- nodo(X), nodo(Y), nodo(Z), reaches_g1(X,Y),
                   in(Y,Z).

:- nodo(U), nodo(V), reaches_g1(U,V), not reaches(U,V).

:~ in(X,Y). [1@1, X,Y]
```