Question 1 and 2 are worth 30% each, and Question 3 is worth 40%. Due Tuesday, November 10, 6pm.

Question 1

A Dung abstract argumentation framework consists of a set A of *arguments* and an *attack* relation $R \subseteq A \times A$ between them.[1] For any two arguments a_1 and a_2, if $(a_1, a_2) \in R$ then we say that a_1 attacks a_2: if one admits argument a_1 then it casts doubts on argument a_2. Computer Scientists are generally interested in finding a subset of arguments that is consistent and that doesn't have any glaring hole. Formally, we s e following two conditions hold: no argument ent outside of E is attacked by an argument fr

Your task is to design an ASP p instance through answer sets. The instance will be provided to your program via two predicates `argument/1` and `attack/2` corresponding to A and R respectively. In the output of your program, you can indicate the chosen subset of argument with a predicate `choose/1` ranging over arguments.

For example, with the following instance:

```
argument(a).
argument(b).
argument(c).
argument(d).
attack(a,b).
attack(b,c).
attack(d,c).
```

a valid output of you

```
choose(a) choose(d)
```


Question 2

Given an undirected graph (V, E), weights on the edges $w : E \to \mathbb{N}$, a target $k \in \mathbb{N}$, and a threshold $O \in \mathbb{N}$, we are interested in finding a k-vertices tree of the graph of weight less than the threshold.[2] In other words, we want to select k vertices and $k-1$ edges from V and E respectively such that they constitute a tree, and the sum of the weights of the selected edges should be below O. That is, F is a feedback edge set iff $(V, E \setminus F)$ is a directed acyclic graph. Develop an ASP program that takes V, E, w, k, and O as input and find a selection of edges satisfying the constraints, or outputs unsatisfiable if the constraints cannot be satisfied. Note that selecting the edges implicitely induces a selection of the vertices, so there is no need for the selected vertices to be explicitely displayed by your program.

An instance to this problem is provided through predicates `vertex/1`, `weight/3`, `target/1`, and `threshold/1`. Any edge has a weight, so statements of the form `weight(a, b, 10).` can be used to declare the existence of an edge between vertices `a` and `b` at the same time as declaring their weight, and there is no need for any redundant `edge/2` predicate. Since the graph is undirected, an edge between two vertices `a` and `b` could also have been declared with `weight(b, a, 10).`. Use the binary predicate `select/2` to indicate which set of edges should be selected.

For example, with the following instance:

```
vertex(v1). vertex(v2). vertex(v3).
vertex(v4). vertex(v5). vertex(v6).
vertex(v7). vertex(v8). vertex(v9).
```

[1] See https://en.wikipedia.org/wiki/Argumentation_framework for more information and references.

[2] See https://en.wikipedia.org/wiki/K-minimum_spanning_tree for more information on this problem and references.

```
weight(v1,v2,3).  weight(v1,v3,3).
weight(v2,v4,1).  weight(v2,v5,5).
weight(v3,v4,3).  weight(v3,v6,4).
weight(v4,v5,4).  weight(v4,v7,1).
weight(v5,v7,7).
weight(v6,v7,2).  weight(v6,v8,2).
weight(v7,v9,3).
weight(v8,v9,2).
target(4).
threshold(4).
```

a valid output of your progr

```
select(v2,v4) select(v
```


Hint: Section 3.1.12 of the Potassco User Guide will prove very useful to understand the syntax for *aggregates* including sum.

Question 3

In this Question, we attempt to write an Answer-Set Program capable of solving puzzles. See https://en.wikipedia.org/wiki/Peg_solitaire for the rules of the game https://pegsolitaire.o

3.1 Explain in Engli he English style and the Eur would use to represent the layout.

3.2 Give an ASP program corresponding to the English layout of th

3.3 An instance of a Peg Solitaire puzzle is given by a layout and an *initial position* (where the pegs are originally located). A solution is a sequence of moves that leads to a final position with a single peg left. Explain in English how you would represent a solution to a given instance. After you explanation, list the ASP predicates you would use in your output to represent the solution.

3.4 Give an ASP program that takes a Peg Solitiare instance as argument and compute a solution. I do not expect your program to be optimized/efficient enough to solve the classical 33-holes English-style puzzle.