German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail

Introduction to Artificial Intelligence, Winter Term 2019
Project 2: $\exists \mathbf{a} \exists \mathbf{s}[\mathbf{Snapped}(\mathbf{IronMan}, \mathbf{Result}(\mathbf{a}, \mathbf{s}))]$

Due: November 21^{st} *at 23:59*

1. Project Description

In this project, you will design and implement a logic-based version of the agent you implemented in Project 1 in a simplified world where *there are no warriors* and *there are only four stones*. A logic-based agent operates by storing sentences about the world in its knowledge base, using an inference mechanism to infer new sentences, and using these sentences to decide which action to take. You must use **Prolog** to implement this project.

To implement this agent correctly, you need to follow the following steps:

a) Implement a Java method `public static void GenGrid (String grid)` to construct logical sentences of the situation calculus representing (i) the starting point of Iron Man, (ii) the location of Thanos, and (iii) the locations of the four infinity stones. The input `grid` is formatted as follows:
`m,n;ix,iy;tx,ty;` $\mathtt{s}_1\mathtt{x},\mathtt{s}_1\mathtt{y},\mathtt{s}_2\mathtt{x},\mathtt{s}_2\mathtt{y},\mathtt{s}_3\mathtt{x},\mathtt{s}_3\mathtt{y},\mathtt{s}_4\mathtt{x},\mathtt{s}_4\mathtt{y}$ where

- `m` and `n` represent the number of rows and number of columns of the grid respectively.
- `ix` and `iy` represent the x (row) and y (column) starting positions of Iron Man.
- `tx` and `ty` represent the x (row) and y (column) positions of Thanos.
- $\mathtt{s}_i\mathtt{x}$,$\mathtt{s}_i\mathtt{y}$ represent the x (row) and y (column) position of stone $\mathtt{s}_i$.

All of the constructed logical sentences should be written to an external file. This file represents part of the agent's knowledge base and should be loaded in the agent's program. You can make `GenGrid` output grids of maximum size 5×5.

b) For each fluent, write a successor-state axiom and add the axioms to the agent's KB.

c) Write a predicate `snapped(S)` and use it to query the agent's KB to generate a plan that Iron Man can follow to collect the four stones, head to the cell where Thanos lies, and snap his fingers. The result of the query should be a situation described as the result of doing some sequence of actions from the initial situation s_0.

2. Sample Input/Output:

Example Input Grid: `5,5;1,2;3,4;1,1,2,1,2,2,3,3`

The following is a visualization of the above grid to make things easier for you.

	0	1	2	3	4
0					
1		S	I		
2		S	S		
3				S	T
4					

Query: `snapped(S)`.
Example Output:
`S = result(snap,result(right,result(collect, result(down,result(right, result(collect,result(right, result(collect,result(down,result(collect, result(left, s0))))))))))`.

3. **Groups:** You may work in groups of *at most* three.

4. **Deliverables**

   a) Source Code
   - A Java class named `Main.java` containing the `GenGrid` method.
   - *Two* output files from `GenGrid` containing logical sentences describing two different grid configurations named as `KB1.pl` and `KB2.pl`.
   - The agent's program loading the output file from `GenGrid`, and containing the implementation of the successor-state axioms. This file should be named `Endgame.pl`.
   - Part of the grade will be on how readable your code is. Use explanatory comments whenever possible

   b) Project Report, including the following.
   - A description of the implementation of `GenGrid`.
   - A discussion of the syntax and semantics of the action terms and predicate symbols you employ.
   - A discussion of your implementation of the successor-state axioms.
   - A description of the predicate `snapped(S)` used to query the KB to generate the plan.
   - At least two running examples from your implementation.
   - If your program does not run, your report should include a discussion of what you think the problem is and any suggestions you might have for solving it.
   - Proper citation of any sources you might have consulted in the course of completing the project. *Under no condition*, you may use on-line code as part of your implementation.

5. **Important Dates**

   Teams. Make sure you submit your team members' details by November 9th at 23:59 using the following link `https://forms.gle/HJ1BiPBFV7wFHJbW7`. Only one team member should submit this for the whole team. Your team members need not be the same team members of Project 1.

   Source code and Report. On-line submission by November 21 at 23:59 using the following link `https://forms.gle/YmB5Nox7XqrTmNQM9`.