German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail
Eng. Nourhan Ehab
Eng. Ammar Yasser

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

Due: December 26^{th} *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 maximum two IMF members to save*. 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 actions to take. You must use **Prolog** to implement this project.

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

a) You will find on the MET website a prolog file `KB.pl` containing a sample KB. Do not add anything to this file, and create a new Prolog file `MissionImpossible.pl` in which you will write your implementation. `MissionImpossible.pl` must be in the same directory in which `KB.pl` lies. Import `KB.pl` at the beginning of `MissionImpossible.pl`.

b) For each fluent, write a successor-state axiom in `MissionImpossible.pl`. You are allowed to use *maximum only two successor-state axioms*. Whenever possible, it is preferable to use built-in predicates rather than defining your own.

c) Write a predicate `goal(S)` and use it to query the agent's KB to generate a plan that Ethan can follow to collect all the IMF members and head to the cell where the submarine lies. You are not required to return an optimal plan. 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.

Important Note: You might write your successor state axioms correctly, yet when you query your KB, your program might run forever. This is because Prolog uses DFS to implement backward chaining, and we know that DFS is incomplete. To solve this issue, consider using the built-in predicate `call_with_depth_limit` (`http://www.swi-prolog.org/pldoc/man?predicate=call_with_depth_limit/3`). This predicate does depth limited search to backchain on the query provided as the first argument of the predicate. You can use this built-in predicate to implement another predicate to do iterative deepening search inside `goal(S)`. In this way, you will guarantee that you will reach a solution since IDS is complete.

2. Sample Input/Output:

Example Input Grid: You will find a sample input KB in `KB.pl`. The first line is a predicate representing Ethan's initial location, the second line is a predicate representing

the initial locations of the IMF members represented as a list of lists of their i, j positions, the third line is a predicate representing the location of the submarine, and the fourth line is a predicate representing the maximum capacity Ethan can carry.

The following is a visualization of the grid in `KB.pl` to make things easier for you.

	0	1	2	3
0	E		S	
1		M	M	
2				
3				

Example Query 1: `goal(S).`
Example Output:

```
S = result(drop, result(up, result(carry, result(down,
result(drop, result(up, result(right, result(carry,
result(down, result(right, s0))))))))).
```

Example Query 2:

```
goal(result(drop, result(up, result(carry, result(down,
result(drop, result(up, result(right, result(carry,
result(down, result(right, s0)))))))))).
```

Example Output:

```
true.
```

Example Query 3:

```
goal(result(up, result(carry, result(down,
result(drop, result(up, result(right, result(carry,
result(down, result(right, s0))))))))).
```

Example Output:

```
false.
```

You have to make sure that your query and output have the exact same format as the above sample runs.

3. **Groups:** You may work in groups of *at most* four. Your teams need not be the same as Project 1's teams.

4. **Deliverables**

   a) Source Code:
   - Rename `MissionImpossible.pl` to `MissionImpossible_XXX.pl` where `XXX` is your team ID. You should only submit this Prolog file.
   - Part of the grade will be on how readable your code is. Use explanatory comments whenever possible.

   b) Project Report, including the following:
   - 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 `goal(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 December 17th at 23:59 using the following link `https://forms.gle/E9f8hQcLSynnihhx8`. 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 Github Classrooms using this invite link `https://classroom.github.com/g/mzb18qKn`.