EECS 3401 Introduction to AI & Logic Programming
Report 4 Specification
Due: Monday, November 21, 2:15pm
Where and how: see Section 1.3

1 Main points

Be sure to read and follow all the guidelines from the links on **reports** and **academic honesty** from the WWW home page for the course.

1.1 Learning objectives

- Instrumenting a Prolog program to analyze its performance
- Practice writing algorithms
- Practice writing report documents

1.1 To hand in

The report consists of two documents.

The first document is **report_4.pl** that is similar to the files you submitted for reports 1 and 2.

The second document is a properly formatted stand-alone report called **report_4.pdf**. For an example structure follow the link *Report → Example for Reports 3, 4 and 5*, where you will find a link to an annotated document describing the general structure of a stand-alone report.

Your **report_4.pl** file should still have suitable comments in it, so one does not have to read the report to be able to use and understand your predicates.

1.2 Electronic submission

Before the deadline, submit the following files – an automatic program will close the submission directory at 2:15pm by the clock in the Prism system. You should submit your work every few days as you add to the report documents. Leaving work to the last day is a poor and inefficient work and learning strategy. If you leave submission to the last moment, you risk a chance that submission will fail, and as consequence, you will receive a failing grade for the report.

- **3401_report_4_CoverPage.pdf –** a copy of the cover page you download from the forum posting with the subject *Report 1 specification* on which you write your name(s) and EECS account number(s)
- **report_4.pdf** – the report that you write (see Section 4)
- **report_4.pl** – the Prolog program you modified.
- **f12_3_Astar.pl** – that you download from the search example programs, and modify as described in section 3
- **f13_10_RTA.pl** – that you download from the search example programs, and modify as described in section 3

To submit your files use the following submit web app (link *File submission* on the course home page). The app requires you to login with your EECS account.

```
https://webapp.eecs.yorku.ca/submit/
```

While you can develop your programs on your personal computer, be sure your files will load and execute correctly on Prism.

2 Mars rover

The Mars rover has to travel from A to B on the surface of Mars, where A is its current location and B is the goal location. The area over which the rover may traverse is divided into a 2-d square grid of squares where the rows and columns of the grid are numbered from 1 up to the maximum grid number G; i.e. the rover roams over a square area. Figure 1 shows what the grid looks like when G = 5 but, of course, the actual G would be much larger.

	1	2	3	4	5
1					
2					
3					
4					
5					

Figure 1: An example Martian grid of grid-size 5.

The rover may only move from one grid-location to the next along either a row or a column. Each grid-location is associated with a number that gives the difficulty of moving into that square from an adjacent square. Fortunately, for the traversable area the difficulty of entering a square is given by the following formula.

$$Difficulty = \mathrm{mod}(row^3 + column^3, 17)$$

The rover's current location A could be any grid-location, and its goal could be any grid-location B such that A ≠ B.

For your information Figures 2 through 4 show the difficulty values for grids of various sizes.

2	9	11	14	7	13
9	16	1	4	14	3
11	1	3	6	16	5
14	4	6	9	2	8
7	14	16	2	12	1
13	3	5	8	1	7

Figure 2: The difficulty values in a Martian grid of size 6x6.

2	9	11	14	7	13	4
9	16	1	4	14	3	11
11	1	3	6	16	5	13
14	4	6	9	2	8	16
7	14	16	2	12	1	9
13	3	5	8	1	7	15
4	11	13	16	9	15	6

Figure 3: The difficulty values in a Martian grid of size 7x7.

2	9	11	14	7	13	4	3
9	16	1	4	14	3	11	10
11	1	3	6	16	5	13	12
14	4	6	9	2	8	16	15
7	14	16	2	12	1	9	8
13	3	5	8	1	7	15	14
4	11	13	16	9	15	6	5
3	10	12	15	8	14	5	4

Figure 4: The difficulty values in a Martian grid of size 8x8.

3 The Task

The given file **report_4.pl** contains instructions to load the files needed for the system, a definition of the predicate **run** that is the start of execution, and some example calls to the system.

In the file **reporty_4.pl** define the predicate *createEdges(GridSize)* that asserts all the edge predicates, *s(State, NewState, Cost)*, between the locations in a square grid of size *GridSize* × *GridSize* that the A* and RTA* use. For this problem you will notice in **report_4.pl** that a state is represented by the compound term *RowNumber–ColumnNumber*.

Define two h predicates *h_e(State, H)* and *h_m(State, H)* to be used by the search algorithms. The predicate *h_e* is to use the Euclidian distance between the robot location and the goal location defined as follows.

$$ED = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2}$$

The predicate *h_m* is to use the Manhattan distance between the robot location and the goal location defined as follows.

$$MD = abs(X1 - X2) + abs(Y1 - Y2)$$

Modify the implementation of the A* algorithm as follows.

- Rename the starting point to be astar, so the algorithm is invoked with the call **astar(Start, Path)**
- Be able to use any h predicate out of an unbounded collection. Do not restrict your program to just **h_e** and **h_m**, as the test program will have different names and your program is expected to run.
- To produce only one solution.
- To find the count of the number of searched-for-nodes. To count searched-for nodes you assert and retract a counter predicate by calling an update_counter predicate, with appropriate arguments, at appropriate places within the algorithms – examples are the files **f12_3_Astar_count.pl**, and **f13_1_IDA_star_count.pl**.

Modify the implementation of the RTA* algorithm as follows.

- Be able to use any h predicate out of an unbounded collection.
- To produce only one solution.
- To find the count of the number of searched-for-nodes.

In your modified files, include comments such as "% New" and "% Remove" to clearly show what you changed. If you have changed the files correctly then the query *t1(P1), t2(P2), t3(P3), t4(P4)* should run.

Create 1 long-path test case for each of four variations as shown in the example calls to *run*, for each of the grid sizes from 5 to 9 – the system runs out of stack with a 10-by-10 grid due to the inefficiency of asserting all the facts for the *s* predicate. The point of that part of the exercise is for you practice developing and describing algorithms.

4 The Report

The report is a proper professional report. For an annotated description of the structure of a report, please look at the file found at the link *An annotated version of the Sequence ADT describing the structure of a stand-alone report* by following the link *Resources* and then the link *Examples potential text questions and exercises*. I also recommend you read the course web page *Reports*.

Your report is to contain a description of what you did for the programming tasks described in Section 3, including a description of how you implemented the counter and why you selected to invoke it at those places.

Your report is to include the data from your test cases, and a comparison of the performance of the four search variations.

4 Grading Scheme

The grade for the report is partitioned into the following parts

- Report – programming 25%
- Report – analysis 25%
- Programming and comments – 25%
- Our test cases – 25%