Project 2 Report

- **A description of the implementation of GenGrid.**

  GenGrid takes the grid string as a parameter. First, we create three files for KB1, KB2 and Endgame. Then, we split the string on the semicolon (;) to obtain the grid's row and column numbers, Ironman's position, Thanos' position and the stones' positions. We create a predicate for the grid size *size(M,N)*, a predicate for the pairs of stone positions stones(L), a predicate for ironman *ironman(X,Y,S,L)* where X & Y are Ironman's current position, S is the situation and L is the list of stones' positions. We write those facts to KB1 or KB2 which are considered our knowledge base for two different grids.

- **A discussion of the syntax and semantics of the action terms and predicate symbols you employ.**

  Predicate Symbols:

```
size(M,N).

This is a fact in the knowledge base. It is composed of two arguments which
represent the number of rows (M) and the number of columns (N) of the grid. It
contains the size of the grid.

thanos(X,Y).

This is a fact in the knowledge base. It represents the position of Thanos in our
grid. It is composed of two arguments: X is the x position of Thanos and Y is the y
position of Thanos.

stones([(S1X,S1Y),(S2X,S2Y),(S3X,S3Y),(S4X,S4Y)]).

This is a fact in the knowledge base. It is a list of pairs containing the initial
x and y positions of all the stones.

ironman(X,Y,s0,[]).

This is a fact in the knowledge base. It represents Ironman's initial position (X
is the x position and Y is the y position) in the initial situation (s0) having an
empty list of collected stones ([]).

ironman(X,Y,S,L).

This is our successor state axiom. Refer to the next section for details.

comparestones(L,(X, Y)).

This is used to check whether a pair represented by x and y positions exists in the
list of pairs L. It is composed of two arguments: L is the list of pairs & (X,Y) is
the pair to examine.
```

```
remove((X,Y),L1,L2).

This is used to remove a pair represented by x and y positions from a list of pairs
L1. L2 represents the resulting list of pairs after successfully removing the pair
(X,Y).
```

Action Terms:

```
collect: A constant which indicates the action of collecting a stone.

up: A constant which indicates the action of moving upwards.

down: A constant which indicates the action of moving downwards.

right: A constant which indicates the action of moving right.

left: A constant which indicates the action of moving left.
```

- **A discussion of your implementation of the successor-state axioms.**

  ironman(X,Y,S,L) :
  X and Y represent the current position of Ironman. S represents the situation resulting from performing a certain action A on the previous situation S1, where A could be any of the action terms mentioned above. L is the list of collected stones so far.

  In case the action is collect, we check if Ironman's position contains a stone using the predicate *comparestones(L,(X,Y))*. If this predicate succeeds, due to backward chaining, we remove the stone from the list of collected stones L to lead to the previous situation where the stone was not collected. The stone is removed from L using the predicate *remove((X,Y), L, L1)*. If this predicate succeeds, a recursive call to *ironman(X,Y,S1,L1)* is made.

  In case of any of the movement actions (up, down, left or right), we check that the previous situation's position (A,B) and the current situation's position (X,Y) are both within the grid borders. The previous situation's position is calculated according to the movement. In the up movement, A is X+1 and B is Y. In the down movement, A is X-1 and B is Y. In the right movement, A is X and B is Y-1. In the left movement, A is X and B is Y+1. We make the recursive call *ironman(A,B,S1,L)*.

- **A description of the predicate snapped(S) used to query the KB to generate the plan.**

  It is composed of one argument (S) which is the final situation representing the goal. We make sure that Ironman is in the same position as Thanos using the predicates thanos(X,Y) and ironman(X,Y,S1,L). L represents the collected stones which is matched with stones(L), a fact in our knowledge base. S is the final result of the action "snap" and the situation S1. S1 is the resulting situation from applying all the actions that would lead to the goal.

- **At least two running examples from your implementation.**

  Grid 1:
  “5,5;1,2;3,4;1,1,2,1,2,2,3,3”

  Knowledge Base Facts:
  size(5,5).
  thanos(3,4).
  stones([(1,1),(2,1),(2,2),(3,3)]).
  ironman(1,2,s0,[]).

  Output:
  result(snap, result(right, result(collect, result(down, result(right, result(collect, result(right, result(collect, result(down, result(collect, result(left, s0))))))))))).

  Grid 2:
  "5,5;1,2;4,2;1,3,2,1,3,3,4,1"

  Knowledge Base Facts:
  size(5,5).
  thanos(4,2).
  stones([(1,3),(2,1),(3,3),(4,1)]).
  ironman(1,2,s0,[]).

  Output:
  result(snap, result(right, result(collect, result(down, result(down, result(collect, result(up, result(left, result(left, result(collect, result(down, result(down, result(collect, result(right, s0))))))))))))).