Name: NIM:

Homework 5
TK2ICM: *Logic Programming* (CSH4Y3)
Second Term 2018-2019

Due date	:	Wednesday, May 1, 2019 at 07:00 a.m. CeLoE time
Type	:	***open all**, individual, cooperation is allowed*

Instruction:

1. This homework is due **Wednesday May 1 at 07:00 a.m. CeLoE time**. Please submit your homework through the submission slot at CeLoE. You are allowed to discuss these problems with other class participants, but make sure that you solve the problems individually. Copying answers from elsewhere without understanding them will not enhance your knowledge.
2. You may use any reference (books, slides, internet) as well as ask other students who are not enrolled to this class.
3. Discussion with fellow participants is encourage as long as you do not copy or rewrite the codes without understanding them. Try solving the problem individually before consulting other students.
4. Try to solve all problem in a time constraint to make you accustomed to the exam condition.
5. Use the predicate name as described in each of the problem. **The name of the predicate must be precisely identical**. Typographical error may lead to the cancellation of your points.
6. Submit your work to the provided slot at CeLoE under the file name `Hw5-<your_name>.pl`. For example: `Hw5-Albert.pl`. Please see an information regarding your nickname at Google Classroom.

1 One Step Movement

Remark 1 This problem is worth 20 points.

Suppose Albert consider the two dimensional space $\mathbb{R}^2$ and he observe all points with integer coordinates, i.e., any point of the form (x, y) where $x, y \in \mathbb{Z}$, e.g.: $(1, 2)$, $(2, -1)$, $(-1, 5)$, etc.. Albert wants to create a Prolog predicate which determines the final position of an object if such object can only move one step in northward, southward, eastward, or westward direction. For clarity, suppose Albert has an object located at point $A\,(3, 3)$ as in Figure 1. The final position of this object after one movement in either northward, southward, eastward, or westward direction is either $A_1\,(3, 4)$, $A_2\,(3, 2)$, $A_3\,(4, 3)$, or $A_4\,(2, 3)$, respectively. Write a predicate `go/3` which takes three arguments `Initial`, `Movement`, and `Final`.

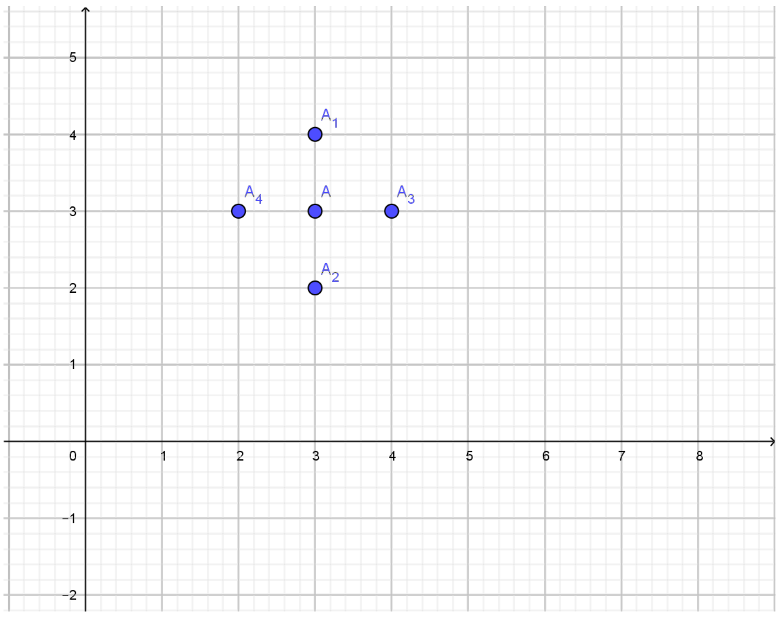

Figure 1: A pictorial description of a one step movement of an object located at $A\,(3, 3)$. The final position is either $A_1\,(3, 4)$, $A_2\,(3, 2)$, $A_3\,(4, 3)$, or $A_4\,(2, 3)$.

`Initial` is the initial coordinate of a point, i.e., $(x, y) \in \mathbb{R}^2$ such that $x, y \in \mathbb{Z}$; `Movement` is either `n`, `s`, `e`, or `w`, which correspondingly denotes the northward, southward, eastward, or westward direction; and `Final` is the final position of the object after one step movement. At least two of the arguments must be instantiated. Some examples:

- `?- go((3,3),n,X).` returns `X = (3,4)`.
- `?- go((3,3),s,X).` returns `X = (3,2)`.
- `?- go((3,3),e,X).` returns `X = (4,3)`.
- `?- go((3,3),w,X).` returns `X = (2,3)`.
- `?- go(X,n,(3,3)).` returns `X = (3,2)`.
- `?- go(X,s,(3,3)).` returns `X = (3,4)`.
- `?- go(X,e,(3,3)).` returns `X = (2,3)`.
- `?- go(X,w,(3,3)).` returns `X = (4,3)`.
- `?- go((1,1),X,(1,2)).` returns `X = n`.

- `?- go((1,1),X,(1,0)).` returns `X = s`.
- `?- go((1,1),X,(2,1)).` returns `X = e`.
- `?- go((1,1),X,(0,1)).` returns `X = w`.
- `?- go((1,1),X,(2,2)).` returns **false**.

2 Final Destination

Remark 2 This problem is worth 20 points and it relates to Problem 1.

Suppose Albert initially has an object at a point (x, y). He then performs a sequence of n one step movement as explained in Problem 1. The sequence of movements is expressed in a list. We are interested to find the final position of such object. For example, in Figure 2 the initial point of the object is $(1, 1)$ and Albert performs ten movements, namely: $east \to east \to east \to north \to west \to north \to west \to west \to south \to east$. The final position of the object is $(2, 2)$.

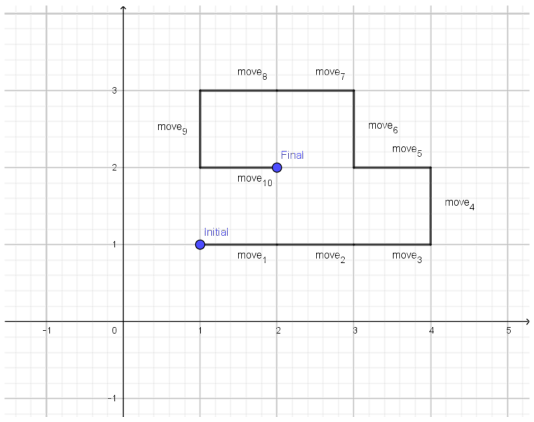

Figure 2: A sequence of 10 movements with initial point $(1, 1)$ and final point $(2, 2)$.

In this problem, your task is to write a predicate `goto/3` which takes three arguments: `+Initial`, `+MoveList`, and `?Final`. The first two arguments, namely `+Initial` and `+MoveList` are always instantiated. `+Initial` describe the initial position of the object while `+MoveList` is a list whose elements are `n`, `s`, `e`, or `w`, `+MoveList` explains the sequence of movements of the object and it is never empty. For example, we have:

- `?- goto((1,1),[s],X).` returns `X = (1,0)`.
- `?- goto((1,1),[e,e,e,n],X).` returns `X = (4,2)`.
- `?- goto((1,1),[e,n,w,s],X).` returns `X = (1,1)`.
- `?- goto((1,1),[e,e,e,n,w,n,w,w,s,e],X).` returns `X = (2,2)`.

3 Final Destination Revisited

Remark 3 This problem is worth 20 points and it relates to Problem 2.

In this problem your task is to extend the predicate `goto/3` in Problem 2 so that it can handle the case whenever the first argument is uninstantiated given the information about the second and the third arguments. In other words, the predicate `goto/3` can determine the original position of the object given the object's movement path and the its final coordinate. For example, we have:

- `?- goto(X,[n],(-1,2)).` returns `X = (-1,1)`.
- `?- goto(X,[s,w,n,e],(3,2)).` returns `X = (3,2)`.
- `?- goto(X,[w,w,w,s],(1,0)).` returns `X = (4,1)`.
- `?- goto(X,[e,e,e,n,w,n,w,w,s,e],(2,2)).` returns `X = (1,1)`.

4 Minimum Movements

Remark 4 This problem is worth 20 points and it relates to Problem 2.

In Problem 2 we have seen that the final point of the object is sometimes reachable using fewer number of movements described by the list. For instance, if the initial point is $(1, 1)$ and the movements is expressed by the list `[e,e,e,n,w,n,w,w,s,e]`, then the final coordinate is $(2, 2)$. This final position is reachable from $(1, 1)$ using two consecutive movements, namely $east \rightarrow north$ (or $north \rightarrow east$). We also infer that it is impossible from $(1, 1)$ to reach $(2, 2)$ using fewer than two movements. Hence, we say that the number of minimum movements from $(1, 1)$ to $(2, 2)$ is two steps. Similarly, it is easy that the number of minimum steps from $(0, 1)$ to $(2, 3)$ is four, one of these movements is $north \rightarrow north \rightarrow east \rightarrow east$.

In this problem your task is to write the predicate `shortdistance/3` consisting of three arguments: `+Initial`, `+Direction`, and `?Steps`. `+Initial` describes the initial point of the object while `+Direction` is the list of `n`, `s`, `e`, or `w` explaining the movement from `+Initial` to a particular point. The predicate returns true if `?Steps` is the number of minimum movements from initial point to some final point. For instance, we have:

- `?- shortdistance((1,1),[n,e,s],D).` returns `D = 1`.
- `?- shortdistance((1,0),[n,e,s,w],D).` returns `D = 0`.
- `?- shortdistance((1,1),[s,e,e,s],D).` returns `D = 4`.
- `?- shortdistance((1,1),[e,e,e,n,w,n,w,w,s,e],D).` returns `D = 2`.

5 Efficient Movements

Remark 5 This problem is worth 20 points and it relates to Problem 4.

In Problem 4 we have constructed the predicate `shortdistance` which determines the number of minimum movements from an initial point to a final point which is obtained by performing a sequence of movements based on a particular list. Here, your task is to construct a predicate `shortpath` to produce any possible efficient movements described by the list. For example, if the initial point is $(1, 1)$ and the movements is described by the list `[e,e,e,n,w,n,w,w,s]`, then the final point is $(0, 3)$. This point is reachable from $(1, 1)$ using three steps movements, namely: `[w,n,n]`, `[n,w,n]`, or `[n,n,w]`.

The movements' list is never empty. In case the initial point and the final point are identical, the predicate returns the string `stay` instead of giving an example of movements' path. Some test cases:

- `?- shortpath((1,1),[e,e,e,n,w,n,w,w,s,e],Path).` returns
```
Path = [e, n] ;
Path = [n, e] ;
```
- `?- shortpath((1,2),[e,e,n,n,n,w,w,w,s],Path).` returns
```
Path = [w, n, n] ;
Path = [n, w, n] ;
Path = [n, n, w] ;
```
- `?- shortpath((0,0),[w,w,w,s,s,n],Path).` returns
```
Path = [w, w, w, s] ;
Path = [w, w, s, w] ;
Path = [w, s, w, w] ;
Path = [s, w, w, w] ;
```
- `?- shortpath((0,-1),[w,w,s,s,s,e,e,e,e,n],Path).` returns
```
Path = [e, e, s, s] ;
Path = [e, s, e, s] ;
Path = [e, s, s, e] ;
Path = [s, s, e, e] ;
Path = [s, e, e, s] ;
Path = [s, e, s, e] ;
```
- `?- shortpath((0,0),[n,s,w,e],Path).`
```
Path = stay.
```