Programming Project #2
Prolog Programming Assignment

Define and test the Prolog predicates described below. Each of your predicates ***must*** have the same name and signature as the examples in each of the ten problems. Your predicates must behave properly on all instances of valid input types.

Your submission should consist of a single source code text file that includes all facts, predicate definitions, and propositions.

Your file should be named `<your_net_id>.prolog`

You may find additional Prolog language help at the following links:

- SWI-Prolog manual
- SWI-Prolog documentation
- Learn Prolog Now!

The Parameter Mode Indicator

Predicate signatures in Prolog are different from function signatures in C++ or Java. A C++ function signature will indicate the data types of a return value as well as the data types of any local function variables referenced in the association function body. For example, `int myFunction(float f, string s)`.

By contrast, a ***parameter mode indicator*** in Prolog gives information about the intended direction in which information carried by a predicate parameter is supposed to flow. Parameter mode indicators are are meta-symbols and not a formal part of the Prolog language but help in explaining intended semantics to the programmer. They are not used in source code itself.

There is no widely accepted agreement on parameter mode indicators in the Prolog community. A list of these symbols adopted by SWI-Prolog can be found in the Reference Manual in Section 4.1 here.

Note: The SWI-Prolog Reference Manual refers to parameters as “arguments”. Technically, predicates have parameters; functions and methods have arguments. I will not deduct points for any homework or exam responses if you call parameters “arguments”.

1) Second Minimum [10 points]

Description:

Define a predicate `secondMin/2` with the signature `secondMin(+List, –Min2)` where `Min2` is the second lowest *unique* valued element in some list of numbers, `List`. If the list has fewer than two unique elements, then your predicate should display the following, "`ERROR: List has fewer than two unique elements.`"

If one more elements of `List` is not a number, then your predicate should use `writeln(+String)` to display the following message for the first encounter of a non-number element, "`ERROR: "element" is not a number.`", where *`element`* is the value of the non-number element.

Your defintion may *not* use the built-in `sort/2` predicate as a helper predicate. However, you may define your own `mySort/2`.

Predicate Signature with Parameter Modes:

```
secondMin(+List, –Min2)
```

Examples:

```
?- secondMin([17,29,11,62,37,53], M2).
M2 = 17

?- secondMin([512], M2).
ERROR: List has fewer than two unique elements.

?- secondMin([7,5.2,3,6,-3.6,9,-2], M2).
M2 = -2

?- secondMin([12,2,b,7], M2).
ERROR: "b" is not a number.

?- secondMin([3,3,3], M2).
ERROR: List has fewer than two unique elements.
```

2) Classify [10 points]

Description:

Define a predicate `classify/3` that takes a list of integers as an parameter and generates two lists, the first containing containing the even numbers from the original list and the second sublist containing the odd numbers from the original list.

Predicate Signature with Parameter Modes:

```
classify(+List, -Even, -Odd)
```

Examples:

```
?- classify([8,7,6,5,4,3], Even, Odd).
Even = [8,6,4]
Odd = [7,5,3]

?- classify([7,2,3,5,8], Even, Odd).
Even = [2,8]
Odd = [7,3,5]

?- classify([-4,11,-7,9,0], Even, Odd).
Even = [-4,0]
Odd = [11,-7,9]

?- classify([5,13,29], Even, Odd).
Even = []
Odd = [5,13,29]

?- classify([], Even, Odd).
Even = []
Odd = []
```

3) Subslice [10 points]

Description:

Design a predicate `subslice/2` that tests if the list in parameter 1 is a contiguous series of elements anywhere within in the list in parameter 2.

Predicate Signature with Parameter Modes:

```
subslice(+List1, +List2)
```

Examples:

```
?- subslice([2,3,4],[1,2,3,4]).
true.

?- subslice([8,13],[3,4,8,13,7]).
true.

?- subslice([3],[1,2,4]).
false.

?- subslice([],[1,2,4]).
true.

?- subslice([1,2,4],[]).
false.
```

4) Shift [10 points]

Description:

Design a predicate `shift/3` that "shifts" or "rotates" a list *N* places to the left. *N* may be a negative number, i.e. rotate to the right. Note that the rotated list should be the same length as the original list.

Predicate Signature with Parameter Modes:

```
shift(+List, +Integer, -List)
```

Examples:

```
?- shift([a,b,c,d,e,f,g,h],3,Shifted).
Shifted = [d,e,f,g,h,a,b,c]

?- shift([1,2,3,4,5],1,Shifted).
Shifted = [2,3,4,5,1]

?- shift([a,b,c,d,e,f,g,h],-2,Shifted).
Shifted = [g,h,a,b,c,d,e,f]
```

5) Luhn Algorithm [10 points]

Description:

Design a predicate `luhn/1` that is an implementation of the Luhn Algorithm and returns `true` if the parameter is an integer that passes the Luhn test and `false` otherwise.

Refer to these resources for a description of the Luhn Algorithm:

- Rosetta Code (Luhn Test of Credit Card Numbers) [link]
- Wikipedia (Luhn Algorithm) [link]

Predicate Signature with Parameter Modes:

```
luhn(+Integer)
```

Examples:

```
?- luhn(799273987104).
true.

?- luhn(49927398717).
false.

?- luhn(49927398716).
true.
```

6) Graph [10 points]

Description:

Design *two* predicates `path/2` and `cycle/1` that determine structures within a graph whose directed edges are encoded with given instances of `edge/2`. For example, `path(x,y)` should evaluate to `true` if a path exists from vertex `x` to vertex `y`, and `false` otherwise. And `cycle(x)` should evaluate to `true` if a cycle exists which includes vertex `x`.

Note: All edges are directional.

Note: Your solution should avoid infinite recursion.

Note: The Knowledge Base of edges in the example below is for explanation only. You are just responsible for the definitions of `path/2` and `cycle/1`. The Knowledge Base edges used for grading will be different.

Predicate Signature with Parameter Modes:

```
path(+Node1, +Node2)
cycle(+Node)
```

Examples:

```
% Knowledge Base
edge(a,b).
edge(b,c).
edge(c,d).
edge(d,a).
edge(d,e).
edge(b,a).

?- path(b,d)
true.

?- path(e,b).
false.

?- path(c,a).
true.

?- cycle(b).
true.

?- cycle(e).
false.
```

10) Clue [20 points]

Four guests (Colonel Mustard, Professor Plum, Miss Scarlett, Ms. Green) attend a dinner party at the home of Mr. Boddy. Suddenly, the lights go out! When they come back, Mr Boddy lies dead in the middle of the table. Everyone is a suspect.

Upon further examination, the following facts come to light:

- Mr Boddy was having an affair with Ms. Green.
- Professor Plum is married to Ms. Green.
- Mr. Boddy was very rich.
- Colonel Mustard is very greedy.
- Miss Scarlett was also having an affair with Mr. Boddy.

There are two possible motives for the murder:

- Hatred: Someone hates someone else if that other person is having an affair with his/her spouse.
- Greed: Someone is willing to commit murder if they are greedy and not rich, *and* the victim is rich.

Part A: Write the above facts and rules in your Prolog program. Use the following names for the people: **`colMustard`**, **`profPlum`**, **`missScarlet`**, **`msGreen`**, **`mrBoddy`**. Be careful about how you encode (or don't encode) symmetric relationships like marriage - you don't want infinite loops!

```
married(X,Y) :- married(Y,X) % INFINITE LOOP
```

Part B: Write a predicate, suspect/2, that determines who the suspects may be, i.e. who had a motive, given a victim.

```
?- suspect(Killer,mrBoddy)
Killer = suspect_name_1
Killer = suspect_name_2
etc.
```

Part C: Add a single fact to your database that will result in there being a unique suspect. Clearly indicate this line in your source comments so that it can be removed/added for grading.

```
?- suspect(Killer,mrBoddy)
Killer = unique_suspect.
```

7) Zebra Puzzle [20 points]

Description:

Design a predicate that solves the following "Zebra Puzzle" https://en.wikipedia.org/wiki/Zebra_Puzzle

The five biggest DJs in the world are going to play in an electronic music festival, each one in a specific stage. They are side by side waiting to play. Find out their nationalities, hobbies and which genre they play.

DJ Properties:

- *Shirt colors*: black, blue, green, red, white
- *Nationalities*: American, Canadian, Dutch, French, Scottish
- *Music genres*: drum and bass, dubstep, EDM, house, trance
- *Performance stages*: Arcadia, Asgard, Shangri-la, Valhalla, Xibalba
- *Ages*: 25, 30, 35, 40, 45
- *Hobbies*: camping, juggling, painting, singing, surfisng

Clues

- The Scott is somewhere to the left of the DJ wearing the White shirt.
- At the fourth position is the DJ who is going to play on the Arcadia stage.
- The 30-year-old DJ is at the first position.
- The DJ that plays EDM is exactly to the right of the Canadian.
- The one who likes Painting is next to the DJ who plays Dubstep.
- The DJ wearing the Black shirt is somewhere between the Scott and the Dubstep player, in that order.
- The French DJ is next to the one wearing the Blue shirt.
- At one of the ends is the DJ that likes Camping.
- The DJ who is going to play on the Asgard stage is wearing the Blue shirt.
- The one that likes Painting is somewhere between the DJ wearing Green and the DJ wearing Blue, in that order.
- At the fifth position is the DJ who plays Drum and bass.

- In the middle is the DJ who is going to play on the Asgard stage.
- The one who plays Trance is next to the one who plays Dubstep.
- The Canadian is exactly to the left of the DJ who likes Juggling
- The DJ whose hobby is Singing is exactly to the right of the DJ wearing the Black shirt.
- The DJ in his Mid-thirties is next to the DJ who is into Juggling.
- The 40-year-old DJ is at the fourth position.
- The 40-year-old DJ is somewhere between the Dutch and the youngest DJ, in that order.
- The DJ wearing Blue is somewhere to the left of the DJ who is going to play on the Xibalba stage.
- The one who enjoys Surfing is going to play on the Valhalla stage.
- The DJ wearing the Red shirt is somewhere to the right of the French.