Assignment2 – Agents

Problem 2.1

For each of the following *agent architectures*, develop a *PEAS* description of the task *environment*.

Additionally, characterize the *environments* of these *agents* according to the *properties* discussed in the *lecture*. Where not "obvious", *justify* your choice with a short *sentence*.

Finally, choose suitable designs for the *agents*.

1. Robot soccer *player*
2. *Internet* book-shop *agent* (that is: an *agent* for book shops that stocks up on books depending on demand)
3. Autonomous Mars rover
4. *Mathematical theorem prover*
5. First-person shooter (Counterstrike, Unreal Tournament etc.)

Problem 2.2

Explain the difference between *agent function* and *agent program*. How many *agent programs* can there be for a given *agent function*?

Problem 2.3

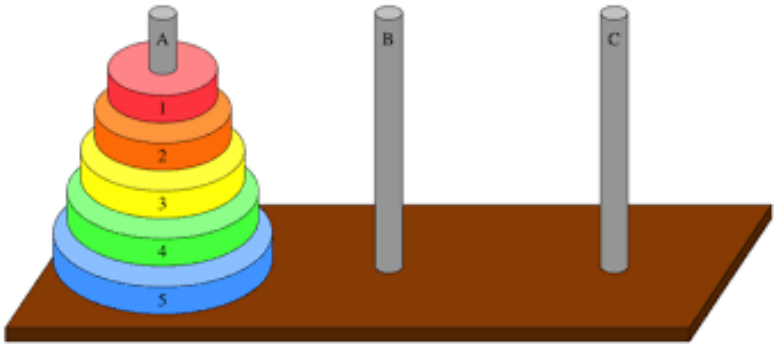
Explain the commonalities of and the differences between the performance measure and the utility function.

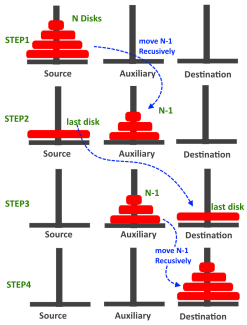
Problem 2.4 (Towers of Hanoi)

The Towers of Hanoi is a puzzle. It consists of three pegs (A, B, and C) and a *number* of disks of different sizes, which can slide onto any peg. The puzzle starts with the disks in a stack in ascending order of size on one peg, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move all disks from peg A to peg B, while obeying the following rules:

1. only one disk can be moved at a time,
2. each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty peg,
3. no larger disk may be placed on top of a smaller disk.

The *idea* of the *algorithm* (for $N > 1$) is to move the top $N - 1$ disks onto the auxiliary peg, then move the bottom disk to the destination peg, and finally moving the remaining $N - 1$ disks from the auxiliary peg to the destination peg.





1. *Write* a *Prolog predicate* that prints out a solution for the Towers of Hanoi puzzle. Use the `write(X)` *predicate* that prints the *value* of *X* (*X* can be simple text or any type of *argument*) to the *screen* and `nl` that prints a new line to write a rule `move(N, A, B, C)` that prints out the solution for moving *N* disks from peg *A* to peg *B*, using *C* as the auxiliary peg.
   Each step of the solution should be of the form "Move top disk from X to Y".
   Examples:

```
?- write(hello), write('␣world!'), nl.
hello world!
true.
```

```
?- move(3, left, center, right).
Move top disk from left to center
Move top disk from left to right
Move top disk from center to right
Move top disk from left to center
Move top disk from right to left
Move top disk from right to center
Move top disk from left to center
true ;
false.
```

2. Determine the *complexity class* of your *algorithm* in terms of the *number* of disks *N* and *explain* how you *computed* it.