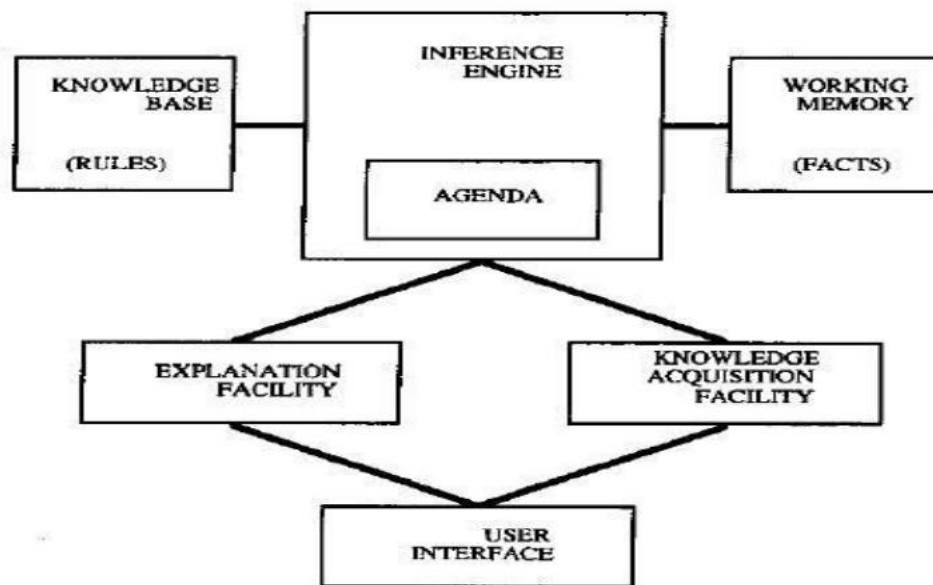


Section 2

CLIPS Language

- CLIPS is a tool for building expert systems.
- C Language Implementation Production System.
- Production system: computer system that relies on facts and rules to guide its decision making.
- Originally developed by the Software Technology Branch (STB) at NASA Johnson Space Center First release in 1986.
- CLIPS is case-sensitive.
- Each CLIPS command must have a matching number of left and right parentheses.

Elements of an Expert System (main components of ES from CLIPS:



- **Knowledge acquisition facility:** automatic way for the user to enter knowledge in the system bypassing the explicit coding by knowledge engineer.
- **Working memory:** global database of facts used by rules.
- **Inference engine:** makes inferences deciding which rules are satisfied and prioritizing.
 - **Agenda:** a prioritized list of rules created by the inference engine, whose patterns are satisfied by facts or objects in working memory.
- **Exploration facility:** explains reasoning of expert system to user.
- **User interface:** mechanism by which user and system communicate

Basic element of an Expert System in clips:

- Fact-list: Global memory for data
- Knowledge-base: Contain all the rules.
- Inference Engine: Control overall execution

Some command in CLIPS:

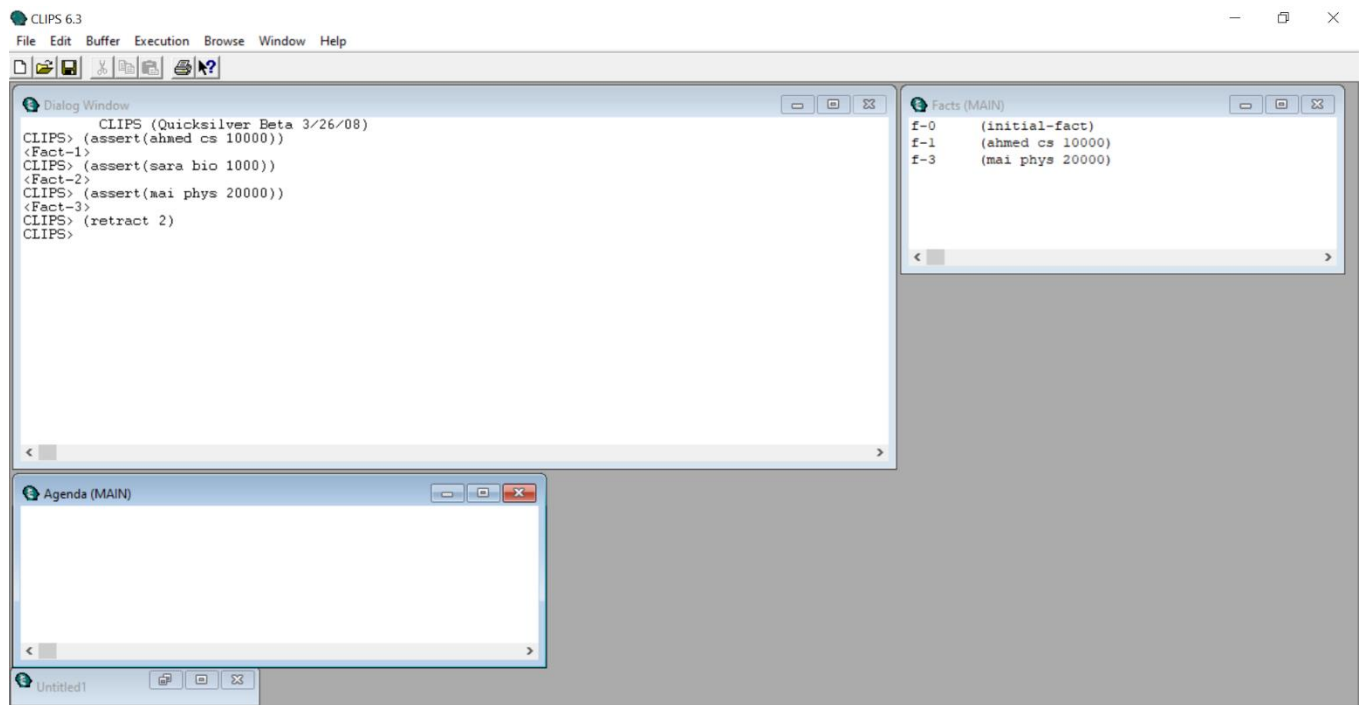
- CLIPS> Commands can be entered directly to CLIPS; this mode is called the top level.
- (exit) → to exit CLIPS program.
- (reset) → is the key method for starting or restarting.
- (run) → to let the program run and applies rules
- (clear) → clears the CLIPS environment and adds the *initialfact-deffacts* to the CLIPS environment.
- (retract <fact-index>+) → to remove fact.

Arithmetic operators:

- Examples:
 - $3 + 4 \rightarrow (+ 3 4)$
 - $(3*4) + (5*6) \rightarrow (+ (* 3 4) (* 5 6))$
 - $(x > 50 \ \& \ y < 30) \rightarrow (and (> x 50) (< y 30))$

Facts

- Fact Assertion
 - (assert<fact>)



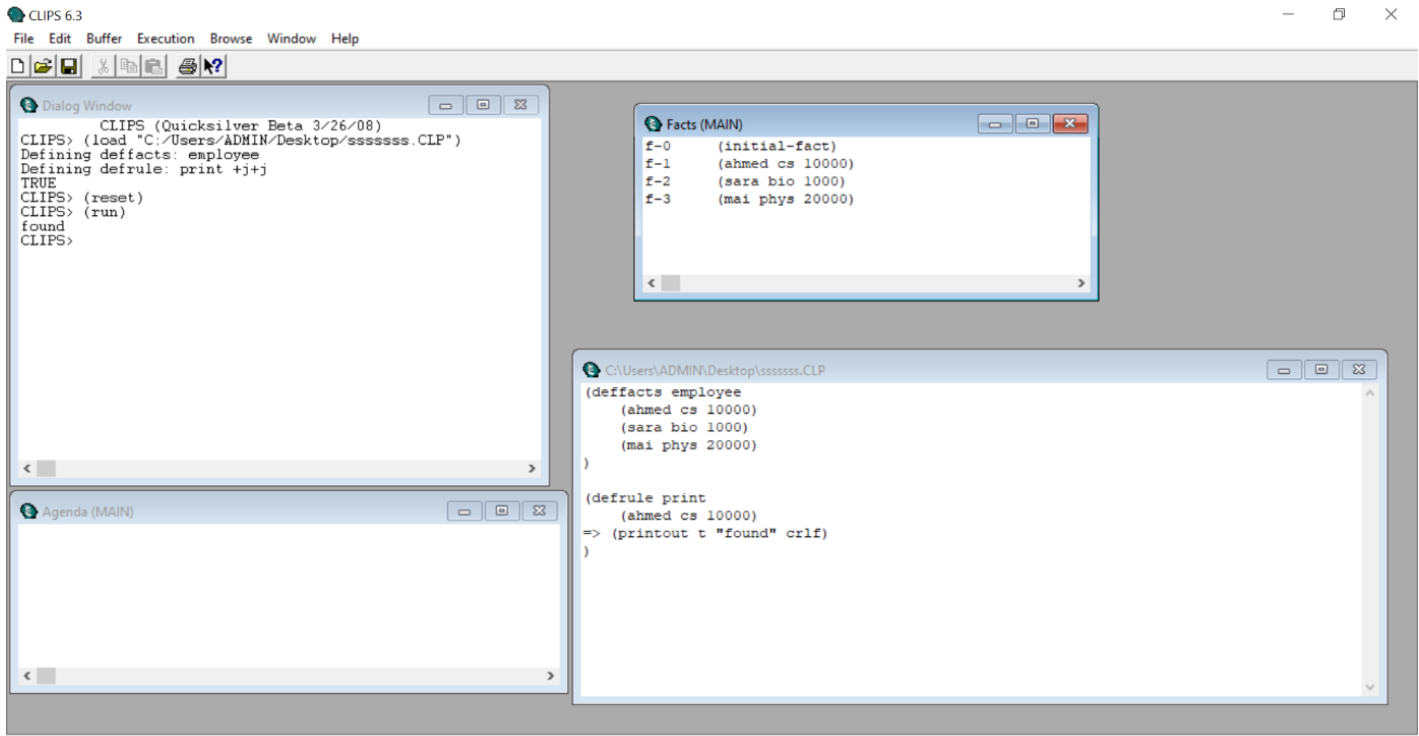
deffacts

- The general format of a deffacts is: (deffacts<deffacts-name>["optional comment"]<facts> *)

```
(deffacts employee
  (em abdullah cs 7300)
  (em hassan bio 2400)
  (em lana phys 50123)
)
```

defrule

- (defrule <rule-name>["comment"]
<pattern>; Left-Hand Side (LHS) of the rule
=>
<actions>; Right-Hand Side (RHS) of the rule



)

Examples:

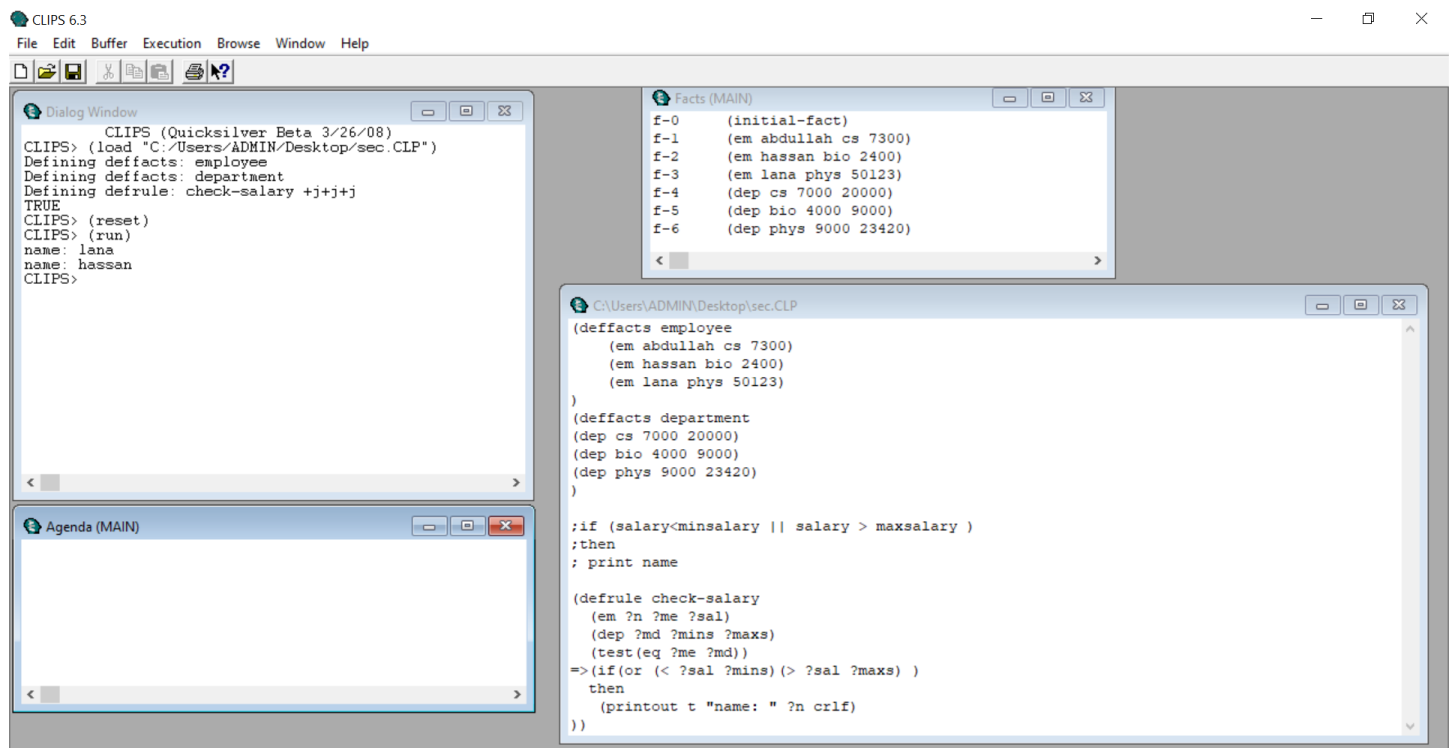
Represents the below facts in Knowledge base and find out how miss the rage of the salary according to employee major.

| Major | Minimum salary | Maximum Salary |
|------------------|----------------|----------------|
| Computer Science | 7000 | 20000 |
| Biology | 4000 | 9000 |
| Physics | 9000 | 23420 |

| Name | Major | Salary |
|----------|------------------|--------|
| Abdullah | Computer Science | 7300 |
| Hassan | Biology | 2400 |
| Lana | Physics | 50123 |

Who has more or less than regular salary the output should be look like this

Answer:



```
CLIPS (Quicksilver Beta 3/26/08)
CLIPS> (load "C:/Users/ADMIN/Desktop/sec.CLP")
Defining deffacts: employee
Defining deffacts: department
Defining defrule: check-salary +j+j+j
TRUE
CLIPS> (reset)
CLIPS> (run)
name: lana
name: hassan
CLIPS>
```

```
f-0 (initial-fact)
f-1 (em abdullah cs 7300)
f-2 (em hassan bio 2400)
f-3 (em lana phys 50123)
f-4 (dep cs 7000 20000)
f-5 (dep bio 4000 9000)
f-6 (dep phys 9000 23420)
```

```
(deffacts employee
(em abdullah cs 7300)
(em hassan bio 2400)
(em lana phys 50123)
)
(deffacts department
(dep cs 7000 20000)
(dep bio 4000 9000)
(dep phys 9000 23420)
)
;if (salary<minsalary || salary > maxsalary )
;then
; print name
(defrule check-salary
(em ?n ?me ?sal)
(dep ?md ?mins ?maxs)
(test(eq ?me ?md))
=>(if(or (< ?sal ?mins)(> ?sal ?maxs) )
then
(printout t "name: " ?n crlf)
))
```

Section 3

Deftemplate

- is used to describe groups of facts sharing the same relation's name and contain common information.
- General format

```
(deftemplate <relation-name> [<optional-comment>]
  <slot-definition>*)
```



```
(slot <slot-name>) | (multislot <slot-name>)
```

```
(deftemplate person "An example deftemplate"
  (slot name)
  (slot age)
  (slot eye-color)
  (slot hair-color))
```

- CLIPS store all template facts known to it in a fact list.
- To add a fact to the list, we use the *assert* command.

```
(deftemplate student
  (slot name)
  (slot age)
  (slot major))

(assert (student (name "John Summers")
                (age 19)
                (major "Information Technology")))
```

- To add a fact to the list, we use *deffacts*.

```
(deftemplate person "person info"
  (slot name (type STRING))
  (slot age (type INTEGER))
  (slot nationality (type STRING))
)

(deffacts person-info
  (person (name "ahmed")
          (age 21)
          (nationality "egyptian"))
)
```

Fields

- Basic data type
- There are eight types of fields, also called the CLIPS primitive data types:
 - FLOAT
 - INTEGER
 - STRING
 - Symbols
 - External address
 - Fact address
 - Instance name
 - Instance address
- CLIPS is case-sensitive.

Variables

- Single field slot
 - ?var
 - ?name (e.g. “amr”)
- Multi field slot
 - \$?var
 - \$?name (e.g., “amr ahmed”)

bind

- Associate symbols
- (bind ?percent (random 1 100))

Example:

CLIPS 2

Section 5

Deffunction

➤ syntax

```
(deffunction <name> [<comment>]
  (<regular-parameter>* [<wildcard-parameter>])
  <action>*>)
```

- < regular-parameter > → <single-field-variable>
- < wildcard-parameter> → <single-field-variable>

Decision Tree

A tree is a hierarchical data structure consisting of:

- Nodes – store information
- Branches – connect the nodes
- The top node is the root, occupying the highest hierarchy.
- The leaves are at the bottom, occupying the lowest hierarchy.
- Every node, except the root, has exactly one parent.
- Every node may give rise to zero or more child nodes.
- A binary tree restricts the number of children per node to a maximum of two.

Examples:

R1: if animal isn't very big and doesn't have squeak

Then animal is squirrel

R2: if animal isn't very big and have squeak

Then animal is mouse

R3: if animal is very big and have long neck

Then animal is giraffe

R4: if animal is very big, doesn't have long neck and have a trunk

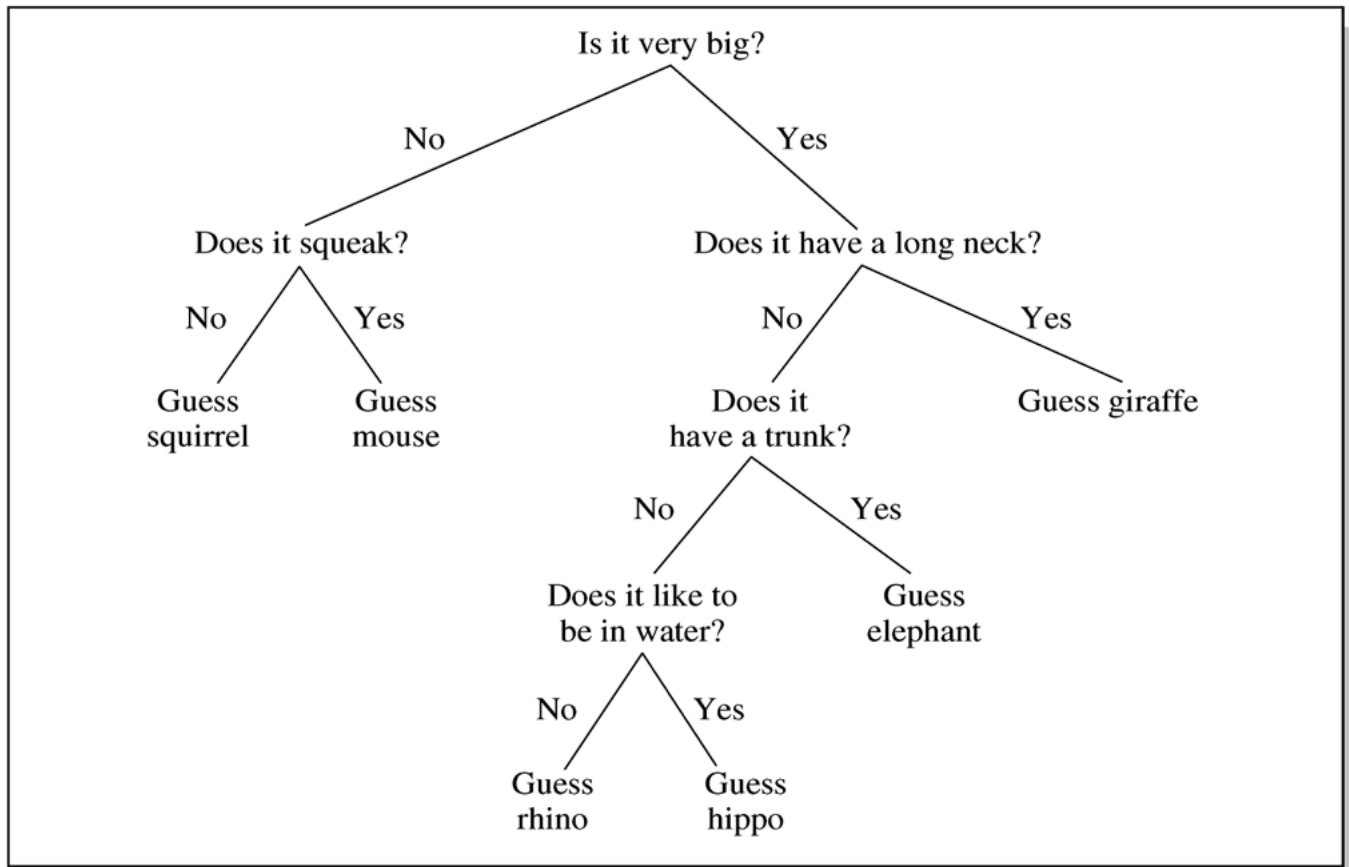
Then animal is elephant

R5: if animal is very big, doesn't have long neck, doesn't have a trunk and it like to be in water

Then animal is hippo

R6: if animal is very big, doesn't have long neck, doesn't have a trunk and doesn't it like to be in water

Then animal is rhino



Examples:

R1: if the car starter is turning and you have petrol

Then call provider

R2: if the car starter is turning and no petrol in car

Then buy petrol

R3: if the car starter is not turning and lights are working and sole is clicked and terminals is clean

Then replace starter

R4: if the car starter is not turning and lights are working and sole is clicked and terminals isn't clean

Then clean terminals

R5: if the car starter is not turning and lights aren't working

Then change battery

R6: if the car starter is not turning and lights are working and sole isn't clicked and fuse isn't working

Then replace fuse

R7: if the car starter is not turning and lights are working and sole isn't clicked and fuse is working

Then replace solenoid

