

This is sanitized output from LogicDemo:

```
c:\temp\>java aima.logic.demos.LogicDemo
```

```
[DPLL]
```

```
numberOfSymbols = 2
clauses are [B, A]
evaluating B
B is not true
At least one clause is unknown
Unsatisfied clauses = 2
clauses are [B, A]
evaluating B
evaluating A
A is not true
At least one clause is unknown
Unsatisfied clauses = 1
clauses are [B, A]
evaluating B
evaluating A
( A AND B ) is (DPLL) satisfiable
numberOfSymbols = 1
clauses are [ ( NOT A ) , A]
evaluating ( NOT A )
( NOT A ) is not true
At least one clause is unknown
Unsatisfied clauses = 2
found unit clause type 2 - assigning
clauses are [ ( NOT A ) , A]
evaluating ( NOT A )
evaluating A
A is not true
( A AND (NOT A) ) is NOT (DPLL) satisfiable
```

PLFCentailsDemo

```
Example from page 220 of AIMA 2nd Edition
KnowledgeBsse consists of sentences
(P => Q)
((L AND M) => P)
((B AND L) => M)
(( A AND P) => L)
((A AND B) => L)
(A)
(B)
Running PLFCentailment on knowledge base with query Q gives
true
```

PLResolutionDemo

```
adding ((B11 => (NOT P11)) AND B11)to knowldegebase
Running plResolution of query (NOT B11) on knowledgeBase
gives false
```

TTentailsDemo

```
(B12 <=> (P11 OR (P13 OR (P22 OR P02))))
(B21 <=> (P20 OR (P22 OR (P31 OR P11))))
(B01 <=> (P00 OR (P02 OR P11)))
(B10 <=> (P11 OR (P20 OR P00)))
(NOT B21)
(NOT B12)
(B10)
(B01)
#
ttentails ("(P00)" ) returns true
#
ttentails ("(NOT P00)" ) returns false
```

WalkSatDemo

```
Example from page 220 of AIMA 2nd Edition
KnowledgeBsse consists of sentences
(P => Q)
((L AND M) => P)
((B AND L) => M)
(( A AND P) => L)
((A AND B) => L)
(A)
(B)
A = true L = true Q = true P = true B = true M = true
```

Forward Chaining Demo 2 -Kings

```
Adding ((King(x) AND Greedy(x)) => Evil(x))
Adding King(John)
Adding King(Richard)
Adding Greedy(John)
Query = Evil(who)
Forward Chaining gives {who=John}
```

Forward Chaining Demo 2 -Missiles

```
..... Adding Rules to KB
Adding ( ((American(x) AND Weapon(y)) AND Sells(x,y,z)) AND
Hostile(z)) => Criminal(x))
Adding ((Missile(m) AND Owns(NoNo,m)) => Sells(West,m,NoNo))
Adding (Missile(missile) => Weapon(missile))
Adding (Enemy(enemy,America) => Hostile(America))
.... Adding Facts
Adding Owns(NoNo, Mone)
Adding American(West)
Adding Enemy(NoNo,America)

Query = Criminal(who)
Forward Chaining gives {who=West}
```