1 Puzzle A

1. $\forall x Baby(x) \implies \neg Rational(x)$
2. $\forall x ManageAnAlligator(x) \implies \neg Loathed(x)$
3. $\forall x \neg Rational(x) \implies Loathed(x)$
4. **(conclusion)** $Baby(Alex) \implies \neg ManageAnAlligator(Alex)$

Converting to clausal form:

1. $\neg Baby(x) \vee \neg Rational(x)$
2. $\neg ManageAnAlligator(x) \vee \neg Loathed(x)$
3. $Rational(x) \vee Loathed(x)$
4. **(conclusion)** $\neg Baby(Alex) \vee \neg ManageAnAlligator(Alex)$

Proof by resolution refutation:
Adding negative of **conclusion** to KB (i.e. sentences 1, 2, 3):

5. $Baby(Alex) \wedge ManageAnAlligator(Alex)$
6. $Baby(Alex)$ 5, And-Elimination
7. $ManageAnAlligator(Alex)$ 5, And-Elimination
8. $\neg Rational(Alex)$ {Alex/x} 1,6 Modus Ponens
9. $Loathed(Alex)$ {Alex/x} 3,8 Modus Ponens
10. $\neg ManageAnAlligator(Alex)$ {Alex/x} 2,9 Modus Ponens
11. NIL 7,10 Modus Ponens

Therefore, the conclusion is **true**.

2 Puzzle B

Assumptions:

1. $\forall x(\exists y(Eats(x, y) \land Pizza(y)) \implies Happy(x))$
2. $\forall x(Foodie(x) \implies \exists y((Pizza(y) \lor Salad(y)) \land Eats(x, y))))$
3. $\forall x(\exists y(Eats(x, y) \land Salad(y)) \implies Healthy(x))$
4. $\forall x(Healthy(x) \implies Gyms(x))$
5. $\forall x(\forall y(Nice(x) \land Happy(y)) \implies \neg Dated(x, y))$
6. $Nice(Ann) \land Foodie(Peter)$

Goal:

$\neg Gyms(Peter) \implies \neg Dated(Ann, Peter)$

Input to Prover9:

```
formulas(assumptions).
all x (exists y (Eats(x, y) & Pizza(y)) -> Happy(x)).
all x (Foodie(x) -> exists y ((Salad(y) | Pizza(y)) & Eats(x, y))).
all x (exists y (Eats(x, y) & Salad(y)) -> Healthy(x)).
all x (Healthy(x) -> Gyms(x)).
all x (all y ((Nice(x) & Happy(y)) -> -Dated(x, y))).
Nice(Ann) & Foodie(Peter).
end_of_list.

formulas(goals).
-Gyms(Peter) -> -Dated(Ann, Peter).
end_of_list.
```

Output from Prover9:

```
============================== Prover9 ===============================
Prover9 (64) version 2009-11A, November 2009.
Process 39550 was started by axy190020 on csgrads1.utdallas.edu,
Fri Mar 12 16:15:02 2021
The command was "LADR-2009-11A/bin/prover9".
============================== end of head ===========================

============================== INPUT =================================

formulas(assumptions).
(all x ((exists y (Eats(x,y) & Pizza(y))) -> Happy(x))).
(all x (Foodie(x) -> (exists y ((Salad(y) | Pizza(y)) & Eats(x,y))))).
(all x ((exists y (Eats(x,y) & Salad(y))) -> Healthy(x))).
(all x (Healthy(x) -> Gyms(x))).
(all x all y (Nice(x) & Happy(y) -> -Dated(x,y))).
Nice(Ann) & Foodie(Peter).
end_of_list.
```

```
formulas(goals).
-Gyms(Peter) -> -Dated(Ann,Peter).
end_of_list.

============================== end of input ==========================

============================== PROCESS NON-CLAUSAL FORMULAS ==========

% Formulas that are not ordinary clauses:
1 (all x ((exists y (Eats(x,y) & Pizza(y))) -> Happy(x))) # label(non_clause).  [assumption].
2 (all x (Foodie(x) -> (exists y ((Salad(y) | Pizza(y)) & Eats(x,y))))) # label(non_clause).  [assumption].
3 (all x ((exists y (Eats(x,y) & Salad(y))) -> Healthy(x))) # label(non_clause).  [assumption].
4 (all x (Healthy(x) -> Gyms(x))) # label(non_clause).  [assumption].
5 (all x all y (Nice(x) & Happy(y) -> -Dated(x,y))) # label(non_clause).  [assumption].
6 Nice(Ann) & Foodie(Peter) # label(non_clause).  [assumption].
7 -Gyms(Peter) -> -Dated(Ann,Peter) # label(non_clause) # label(goal).  [goal].

============================== end of process non-clausal formulas ===

============================== PROCESS INITIAL CLAUSES ===============

% Clauses before input processing:

formulas(usable).
end_of_list.

formulas(sos).
-Eats(x,y) | -Pizza(y) | Happy(x).  [clausify(1)].
-Foodie(x) | Salad(f1(x)) | Pizza(f1(x)).  [clausify(2)].
-Foodie(x) | Eats(x,f1(x)).  [clausify(2)].
-Eats(x,y) | -Salad(y) | Healthy(x).  [clausify(3)].
-Healthy(x) | Gyms(x).  [clausify(4)].
-Nice(x) | -Happy(y) | -Dated(x,y).  [clausify(5)].
Nice(Ann).  [clausify(6)].
Foodie(Peter).  [clausify(6)].
-Gyms(Peter).  [deny(7)].
Dated(Ann,Peter).  [deny(7)].
end_of_list.

formulas(demodulators).
end_of_list.

============================== PREDICATE ELIMINATION =================

Eliminating Eats/2
8 -Foodie(x) | Eats(x,f1(x)).  [clausify(2)].
9 -Eats(x,y) | -Pizza(y) | Happy(x).  [clausify(1)].
Derived: -Foodie(x) | -Pizza(f1(x)) | Happy(x).  [resolve(8,b,9,a)].
10 -Eats(x,y) | -Salad(y) | Healthy(x).  [clausify(3)].
Derived: -Salad(f1(x)) | Healthy(x) | -Foodie(x).  [resolve(10,a,8,b)].

Eliminating Foodie/1
11 Foodie(Peter).  [clausify(6)].
12 -Foodie(x) | Salad(f1(x)) | Pizza(f1(x)).  [clausify(2)].
Derived: Salad(f1(Peter)) | Pizza(f1(Peter)).  [resolve(11,a,12,a)].
13 -Foodie(x) | -Pizza(f1(x)) | Happy(x).  [resolve(8,b,9,a)].
Derived: -Pizza(f1(Peter)) | Happy(Peter).  [resolve(13,a,11,a)].
14 -Salad(f1(x)) | Healthy(x) | -Foodie(x).  [resolve(10,a,8,b)].
Derived: -Salad(f1(Peter)) | Healthy(Peter).  [resolve(14,c,11,a)].

Eliminating Healthy/1
15 -Salad(f1(Peter)) | Healthy(Peter).  [resolve(14,c,11,a)].
```

```
16 -Healthy(x) | Gyms(x).  [clausify(4)].
Derived: -Salad(f1(Peter)) | Gyms(Peter).  [resolve(15,b,16,a)].

Eliminating Nice/1
17 Nice(Ann).  [clausify(6)].
18 -Nice(x) | -Happy(y) | -Dated(x,y).  [clausify(5)].
Derived: -Happy(x) | -Dated(Ann,x).  [resolve(17,a,18,a)].

Eliminating Gyms/1
19 -Salad(f1(Peter)) | Gyms(Peter).  [resolve(15,b,16,a)].
20 -Gyms(Peter).  [deny(7)].
Derived: -Salad(f1(Peter)).  [resolve(19,b,20,a)].

Eliminating Dated/2
21 -Happy(x) | -Dated(Ann,x).  [resolve(17,a,18,a)].
22 Dated(Ann,Peter).  [deny(7)].
Derived: -Happy(Peter).  [resolve(21,b,22,a)].

Eliminating Salad/1
23 -Salad(f1(Peter)).  [resolve(19,b,20,a)].
24 Salad(f1(Peter)) | Pizza(f1(Peter)).  [resolve(11,a,12,a)].
Derived: Pizza(f1(Peter)).  [resolve(23,a,24,a)].

Eliminating Pizza/1
25 Pizza(f1(Peter)).  [resolve(23,a,24,a)].
26 -Pizza(f1(Peter)) | Happy(Peter).  [resolve(13,a,11,a)].
Derived: Happy(Peter).  [resolve(25,a,26,a)].

Eliminating Happy/1
27 Happy(Peter).  [resolve(25,a,26,a)].
28 -Happy(Peter).  [resolve(21,b,22,a)].
Derived: $F.  [resolve(27,a,28,a)].

============================== end predicate elimination =============

Auto_denials:  (no changes).

Term ordering decisions:
Predicate symbol precedence:  predicate_order([ ]).
Function symbol precedence:  function_order([ ]).
After inverse_order:  (no changes).
Unfolding symbols: (none).

Auto_inference settings:
  % set(neg_binary_resolution).  % (HNE depth_diff=0)
  % clear(ordered_res).  % (HNE depth_diff=0)
  % set(ur_resolution).  % (HNE depth_diff=0)
    % set(ur_resolution) -> set(pos_ur_resolution).
    % set(ur_resolution) -> set(neg_ur_resolution).

Auto_process settings:  (no changes).


============================== PROOF =================================

% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 29.
% Level of proof is 8.
% Maximum clause weight is 0.000.
% Given clauses 0.

1 (all x ((exists y (Eats(x,y) & Pizza(y))) -> Happy(x))) # label(non_clause).  [assumption].
2 (all x (Foodie(x) -> (exists y ((Salad(y) | Pizza(y)) & Eats(x,y))))) # label(non_clause).  [assumption].
```

```
3 (all x ((exists y (Eats(x,y) & Salad(y))) -> Healthy(x))) # label(non_clause).  [assumption].
4 (all x (Healthy(x) -> Gyms(x))) # label(non_clause).  [assumption].
5 (all x all y (Nice(x) & Happy(y) -> -Dated(x,y))) # label(non_clause).  [assumption].
6 Nice(Ann) & Foodie(Peter) # label(non_clause).  [assumption].
7 -Gyms(Peter) -> -Dated(Ann,Peter) # label(non_clause) # label(goal).  [goal].
8 -Foodie(x) | Eats(x,f1(x)).  [clausify(2)].
9 -Eats(x,y) | -Pizza(y) | Happy(x).  [clausify(1)].
10 -Eats(x,y) | -Salad(y) | Healthy(x).  [clausify(3)].
11 Foodie(Peter).  [clausify(6)].
12 -Foodie(x) | Salad(f1(x)) | Pizza(f1(x)).  [clausify(2)].
13 -Foodie(x) | -Pizza(f1(x)) | Happy(x).  [resolve(8,b,9,a)].
14 -Salad(f1(x)) | Healthy(x) | -Foodie(x).  [resolve(10,a,8,b)].
15 -Salad(f1(Peter)) | Healthy(Peter).  [resolve(14,c,11,a)].
16 -Healthy(x) | Gyms(x).  [clausify(4)].
17 Nice(Ann).  [clausify(6)].
18 -Nice(x) | -Happy(y) | -Dated(x,y).  [clausify(5)].
19 -Salad(f1(Peter)) | Gyms(Peter).  [resolve(15,b,16,a)].
20 -Gyms(Peter).  [deny(7)].
21 -Happy(x) | -Dated(Ann,x).  [resolve(17,a,18,a)].
22 Dated(Ann,Peter).  [deny(7)].
23 -Salad(f1(Peter)).  [resolve(19,b,20,a)].
24 Salad(f1(Peter)) | Pizza(f1(Peter)).  [resolve(11,a,12,a)].
25 Pizza(f1(Peter)).  [resolve(23,a,24,a)].
26 -Pizza(f1(Peter)) | Happy(Peter).  [resolve(13,a,11,a)].
27 Happy(Peter).  [resolve(25,a,26,a)].
28 -Happy(Peter).  [resolve(21,b,22,a)].
29 $F.  [resolve(27,a,28,a)].

============================== end of proof ==========================

============================== STATISTICS ============================

Given=0. Generated=1. Kept=0. proofs=1.
Usable=0. Sos=0. Demods=0. Limbo=0, Disabled=22. Hints=0.
Kept_by_rule=0, Deleted_by_rule=0.
Forward_subsumed=0. Back_subsumed=0.
Sos_limit_deleted=0. Sos_displaced=0. Sos_removed=0.
New_demodulators=0 (0 lex), Back_demodulated=0. Back_unit_deleted=0.
Demod_attempts=0. Demod_rewrites=0.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=0. Nonunit_bsub_feature_tests=0.
Megabytes=0.04.
User_CPU=0.01, System_CPU=0.00, Wall_clock=0.

============================== end of statistics =====================

============================== end of search =========================

THEOREM PROVED

Exiting with 1 proof.

Process 39550 exit (max_proofs) Fri Mar 12 16:15:02 2021
```

Therefore, the conclusion is **true**.