Robin Mehta
AI Homework 3

A	B	C	A and B	A and B → C	A → C	B → C	(A→C) V (B→C)	A V B	- (A and B)	-C	-A V –B V- C	(A V B) and (-C V –D V –E)
1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	0	1	0	0	0	0	1	0	1	1	1
1	0	0	0	1	0	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	0	0	1	1	0
0	0	1	0	1	1	1	1	0	0	0	0	0
0	1	1	0	1	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
0	1	0	0	1	1	0	1	1	1	1	1	1

Problem 1, Part 1)

a. True
b. True
c. False
d. True
e. True
f. True

Problem 1, Part 2)

a. Prove bi-directionally:
  1) If True = true, which means that alpha is valid in every model.
  2) If alpha is valid (true in all models), True is still true in every model.

b. Prove alpha entails beta iff (alpha → beta) is valid
  1) In every model where alpha is true, beta is also true
  2) If (alpha → beta) is valid (true in all models), beta will also be true.

c. Prove alpha ≡ beta iff (alpha ←→ beta)

  1) A truth table proves that where (alpha ←→ beta), alpha ≡ beta is true in all models.
  2) A truth table proves that where (alpha ≡ beta), alpha → beta is also true in all models.

d. Prove alpha entails beta iff (alpha ∧ ~beta) is unsatisfiable.
  1) Let (alpha ∧ ~beta) be satisfiable (at least one model exists where alpha is true and beta is false), which does not show entailment.
  2) Let (alpha entails beta) be valid. Here, there is never a case where alpha is true and beta is false.

Overall, if alpha is true and beta is never false, (alpha ∧ ~beta) is unsatisfiable.

Robin Mehta
AI Homework 3

Problem 2, Part 1)

A
~C
~A V B
C V D
E
~B V ~D V ~E V F

Overall:

(A) ∧ (~C) ∧ (~A V B) ∧ (C V D) ∧ (E) ∧ (~B V ~D V ~E V F)

Problem 2, Part 2) Use resolution to prove F

Assume: F is false. Prove using contradiction

~F → (~B V ~D V ~E) :

~F → ~B must be true

true → ~(~A V B), given A, meaning B is true, meaning ~B is false. (true → false is false). Contradiction.

OR

~F → ~D must be true

true → (C V D), given C, meaning D is true, meaning ~D is false. (true → false is false). Contradiction.

OR

~F → ~E must be true

true → ~E, given E is true, meaning ~E is false. (true → false is false). Contradiction.

Finally:

~F → false or false or false

true → false. Contradiction!

Robin Mehta
AI Homework 3

Problem 3, Part 1: Determine whether the pairs can be unified

1. No. Since B and C are different constants, and y and y are the same variable, these pairs cannot be unified.
2. No. B is not equal to GameOf(x), meaning that B, being a constant, cannot be substituted for GameOf(x).
3. Yes. C(B, FriendOf(B), GameOf(FriendOf(B)))
4. Yes. P(A, FriendOf(B))
5. No. By the unique names assumption, two pairs of different functions cannot be unified.
6. No. By the unique names assumption, A and C are different constants, and so FriendOf(x,x) cannot be substituted with FriendOf(A,C).

Problem 4

Part 1: Translate the knowledge base into first-order-logic (using quantifiers!)

1) Student(Pat)
2) Restaurant(Ayse)
3) GetsScholarship(Pat)
4) Vx(Student(x) ∧ GetScholarship(x)) → SpecialOccasion(x)
5) Ex(Patron(Pat, y))
6) Vx,y,z(Student(x) ∧ Patron(x, y) ∧ Restaurant(y) ∧ SpecialOccasion(x)) → Eats(x, z=lentil soup)

Part 2: Convert To CNF

1) Student(Pat) ∧
2) Restaurant(Ayse) ∧
3) GetsScholarship(Pat) ∧
4) ~Student(x) V ~GetsScholarship(x) V SpecialOccasion(x) ∧
5) Patron(Pat, y) ∧
6) ~Student(x) V ~Patron(x, y) V ~Restaurant(y) V ~SpecialOccasion(x)) V Eats(x, z=lentil soup)

Part 3: Prove that Pat eats lentil soup using resolution refutation.

Assume: ~Eats(x=Pat,z=lentil soup)

Given by CNF:

If ~Eats(x=Pat,z=lentil soup) is true, then:

~SpecialOccasion(x) must be true

Given: Student(Pat), GetsScholarship(Pat)

~Student(Pat) V ~GetsScholarship(Pat) V SpecialOccasion(Pat) must be true

false V false V SpecialOccasion(Pat) must be true, so SpecialOccasion(Pat) is true. Contradiction.

OR

~Restaurant(y) must be true

Given: Restaurant(y) is true. Contradiction.

OR

~Patron(x, y) must be true

Given: Patron(Pat, y) is true. Contradiction.

OR

~Student(x) must be true

Given: Student(Pat) is true. Contradiction.

Finally:

~Student(x) V ~Patron(x, y) V ~Restaurant(y) V ~SpecialOccasion(x)) V Eats(x, z=lentil soup)

False V False V False V False V False. Contradiction!

Problem 5:

Part 1: List unary constraints, if any:

1) None
2) Constraints i, ii, iii, iv, v, vi

Part 2:

1) Possible ordered list of the deliverables:

a) BDFHJCK violates constraint iii.

Robin Mehta
AI Homework 3

b) CJBHDKF violates constraint i.

c) HBDFJCK does not violate any constraints!

d) HBDFKJC violates constraint v.

e) HJDBFKC violates constraint iv.

1) Domains after applying arc consistency:

B: 1,2,3,4,5,6

C: 1,2,3,4,5,6,7

D: 2,3,4,5,6,7

E: 1,2,3,4,5,6,7

F: 1,2,3,4,5,6,7

H: 1,2,3,4,5,6

J: 1,2,3,4,5,6,7

K: 2,3,4,5,6,7

3) Backtracking

	B	C	D	F	H	J	K
Initial	1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7	1, 2, 3, 4, 5, 6, 7
B=1	1	3, 4, 5, 6, 7	2	3, 4, 5, 6, 7	3, 4, 5, 6, 7	3, 4, 5, 6, 7	3, 4, 5, 6, 7
D=2	1	3, 4, 5, 6, 7	2	3, 4, 5, 6, 7	Backtrack		
B=2	2	1, 3, 4,	1, 3	1, 3, 4,	1, 3, 4, 5,	1, 3, 4,	3, 4, 5, 6,

		5, 6, 7		5, 6, 7	6, 7	5, 6, 7	7
D=1	2	3, 4, 5, 6, 7	1	3, 4, 5, 6, 7	Backtrack		
D=3	2	1, 4, 5, 6, 7	3	1, 3, 4, 5, 6, 7	1	1, 4, 5, 6, 7	1, 4, 5, 6, 7
H=1	2	4, 5, 6, 7	3	4	1	4, 5, 6, 7	4, 5, 6, 7
F=4	2	5, 6, 7	3	4	1	5, 6, 7	5, 6, 7
C=5	2	5	3	4	1	6, 7	6, 7
J=6	2	5	3	4	1	6	Backtrack
C=6	2	6	3	4	1	5, 7	5, 7
J=5	2	6	3	4	1	5	7
K=7	2	6	3	4	1	5	7

Problem 6, Part 1:

	4	5	6
A	1, 4, 7, or 9	2	1, 4, 7, or 9
B	3	1, 4, 7, or 9	5
C	8	1, 4, 7, or 9	6

2) Reduce the domain for variables: List the new domains.

	4	5	6

Robin Mehta
AI Homework 3

A	4, or 9	2	1, 4, or 7
B	3	4, or 7	5
C	8	7, or 9	6

3) Minimum-remaining value heuristic:
We would consider A4, B5, or C5 since these have 2 values left to consider each.

Part 2: Code

Robin Mehta
AI Homework 3

robin_mehta_solver.py

```
def ac3(sudoku):
    numQueue = list(sudoku)
    box = sudoku[0][0]
    for (x,y), value in np.ndenumerate(sudoku):
        if (value == box):
            solved = True
    solved_sudoku = np.copy(sudoku)
    return False, solved_sudoku

'''
Backtracking search Algorithm
Input: 2D numpy matrix
Output: return True if a solution is found, with solved sudoku, False
'''
def check(sudoku, box, x, y):
    for (a,b), value in np.ndenumerate(sudoku):
        if (value != box and x == a and y == b):
            if (x == a or y == b):
                return False
    return True

def bts(sudoku):
    solved = False
    for (x,y), value in np.ndenumerate(sudoku):
        if(value == "*"):
            solved = False
    solved_sudoku = np.copy(sudoku)
    return solved, solved_sudoku

'''
Main function
'''
def main():
    sudoku_list = read_sudoku(sys.argv[1])
    solved_sudokus = []
    for sudoku in sudoku_list:
        print_sudoku(sudoku)
        if sys.argv[2] == 'ac3':
            print 'Using AC-3'
```