Model Solution, XYZZY@umbc.edu
CMSC 471 HW4 Spring 2018

Total points: 100

1. Checking Validity (10 points, 3 each for 1.2-1.6)

1. P V ¬P

```
    >>> tt_true(expr('P | ~P'))
    True
```

2. P → P

```
tt_true(expr('P ==> ~P'))
False
```

3. P → (P V Q)

```
tt_true(expr('P ==> (P | Q)'))
True
```

4. (P V Q) → P

```
tt_true(expr('(P | Q) ==> P'))
False
```

5. ((A ∧B) →C) ↔ (A → (B → C))

```
tt_true(expr('((A & B) ==> C) <=> (A ==> (B ==> C))'))
True
```

6. ((A → B) → A) → A

```
tt_true(expr('((A ==> B) ==> A) ==> A'))
True
```

2. Satisfiability (9 points, 3 each)

1. P ∧ Q

```
    >>> dpll_satisfiable(expr('P & Q'))
    {P: True, Q: True}
```

2. ALIVE→ ¬DEAD ∧ ¬ALIVE∧ ¬DEAD

```
dpll_satisfiable(expr('ALIVE ==> ~DEAD & ~ALIVE
& ~DEAD'))
```

```
{ALIVE: False, DEAD: False}
```

3. P → ¬ P v P

```
dpll_satisfiable(expr('P ==> ( ~P | P )'))
```

```
{P: True}
```

4. ~ (P v ¬ P)

```
dpll_satisfiable(expr('~(P | ~P)'))
False
```

3. Propositional Consequence (24 points, 3 each)

1. P ∧ Q ⊨ P

```
    True
```

2. p |= p ^ q

```
False
```

3. p |= p v q

```
True
```

4. p |= ~~p

```
True
```

5. p-->q |= ~p-->~q

```
False
```

6. ~p |= p-->q

```
True
```

7. ~q |= p-->q

False

8. p, p-->q |= q

True

9. ~p, q-->p |= ~q

True

4. English to FOL (30 points, 3 each)

There are usually several reasonable ways to express natural language sentences in logic. One source of variation is what to leave implicit and what to make explicit, e.g., omitting obvious types as in rendering 'every person loves their mother' as Ax mother(x,y) => loves(y,x). Another comes from the application of standard tautologies, e.g., you can encode 'no man is an island' as ~Ex man(x) ^ island(x) or as Ax man(x) => ~island(x).

1 . Everything is either dead or alive.

```
Ax dead(x) v alive(x)
```

2 . Zombies are not alive but they are animate

```
Ax zombie(x) => ~alive(x) ^ animate(x)
```

3 . Good food is not cheap and cheap food is not good.

```
Ax (food(x)^good(x) => ~cheap(x)) ^
   (food(x)^cheap(x) => ~good(x))
```

4 . John has exactly two brothers.

```
Ex,y brother(john, x) ^ brother(john, y) ^ ~ x=y
  ^ (Az brother(john, z) => z=x v z=y)
```

5 . No person can have two mothers.

```
Ax person(x) => ~ Ey,z mother(x,y) ^ mother(x,z)
  ^ ~y=z
```

6 . If John has a sister, she is smart.

```
Ax sister(john,x) => smart(x)
```

7. Every person is either male or female and no person can be both male and female.

```
Ax person(x) => (male(x) v female(x)) ^
  ~(male(x) ^ female(x))
```

8. The enemy of your enemy is your friend.

```
Ax,y,x enemy(x, y) ^ enemy(y, z) => friend(x, z)
```

9. An ancestor of your ancestor is your ancestor.

```
Ax,y,z ancestor(x, y) ^ ancestor(y, z) =>
  ancestor(z, z)
```

5. Solving a puzzle with SAT (22 points)

Which answer in this list below is the correct answer to this question?

1. All of the below.
2. None of the below.
3. All of the above.
4. One of the above.
5. None of the above.
6. None of the above.

Explain your reasoning by (a) mapping the problem into propositional logic and (b) showing how the AIMA code can be used to solve this problem.

Hint: These puzzles are hard for people because they are self-referential. Here's a hint. Try associating a propositional variable with each answer, e.g., A1, A2, ...A6. Now write logical sentences that meaning of each answer in terms of the meanings of the other answers. Put these sentences into the form used by AIMA and use the dpll_satisfiable function to see if the result is satisfiable. If you've done things correctly, the sentences will be satisfiable with one of the Ai variables True and the rest False.

```
#5, None of the above, is the only correct answer
```

```
dpll_satisfiable(expr(
    '(A1 <=> A2 & A3 & A4 & A5 & A6) & \
     (A2 <=> ~A3 & ~A4 & ~A5 & ~A6) & \
     (A3 <=> A1 & A2) & \
     (A4 <=> A1 | A2 | A3) & \
     (A5 <=> ~A1 & ~A2 & ~A3 & ~A4) & \
     (A6 <=> ~A1 & ~A2 & ~A3 & ~A4 & ~A5)'))
```

```
{A1: False, A2: False, A3: False, A4: False, A5:
True, A6: False}
```