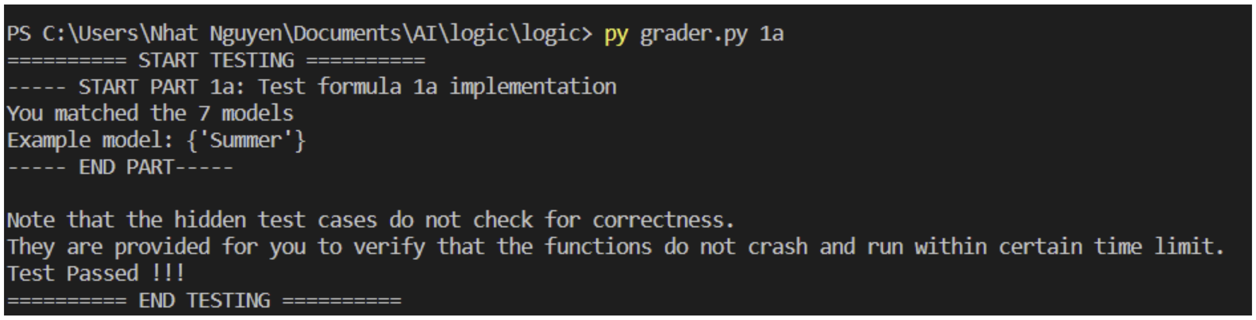
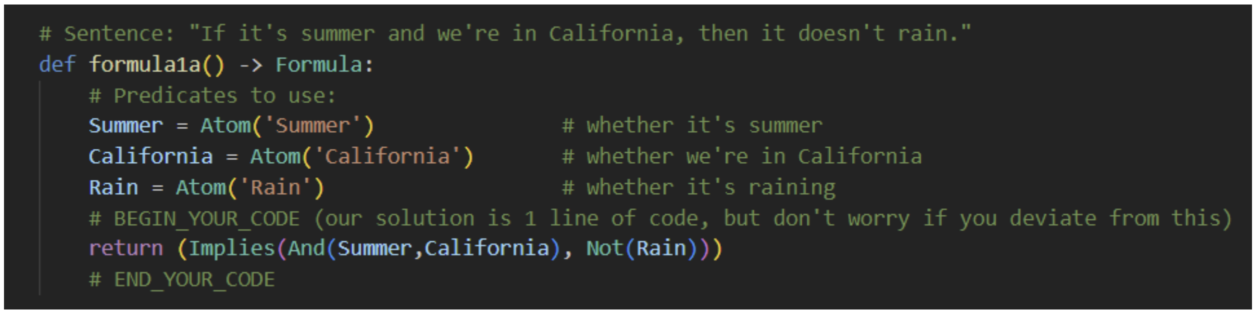
Nhat Nguyen
2-8-2024
Part 1
1a



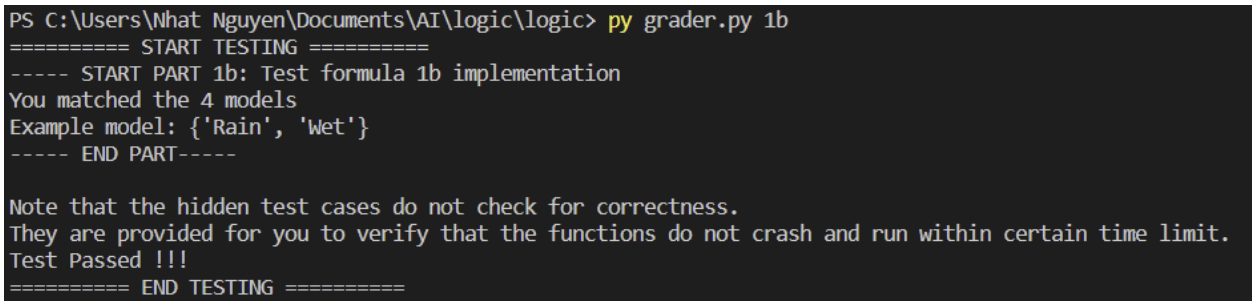
```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 1a
========== START TESTING ==========
----- START PART 1a: Test formula 1a implementation
You matched the 7 models
Example model: {'Summer'}
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

Code:



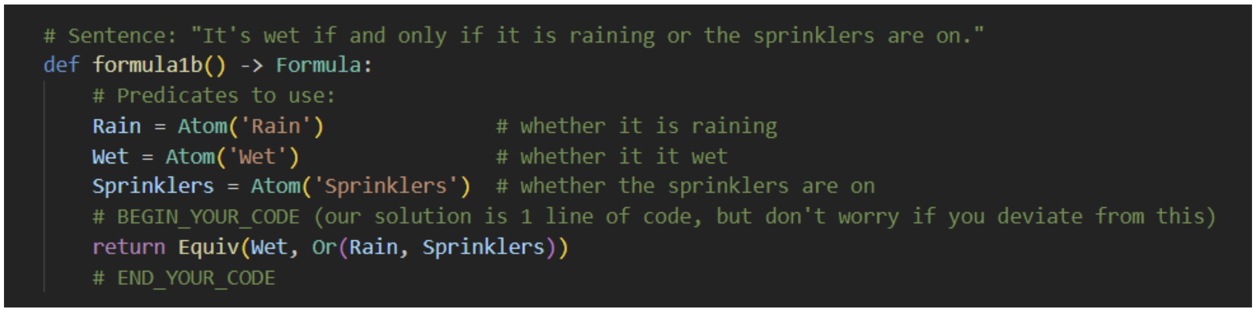
```python
# Sentence: "If it's summer and we're in California, then it doesn't rain."
def formula1a() -> Formula:
    # Predicates to use:
    Summer = Atom('Summer')               # whether it's summer
    California = Atom('California')       # whether we're in California
    Rain = Atom('Rain')                   # whether it's raining
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    return (Implies(And(Summer,California), Not(Rain)))
    # END_YOUR_CODE
```

1b



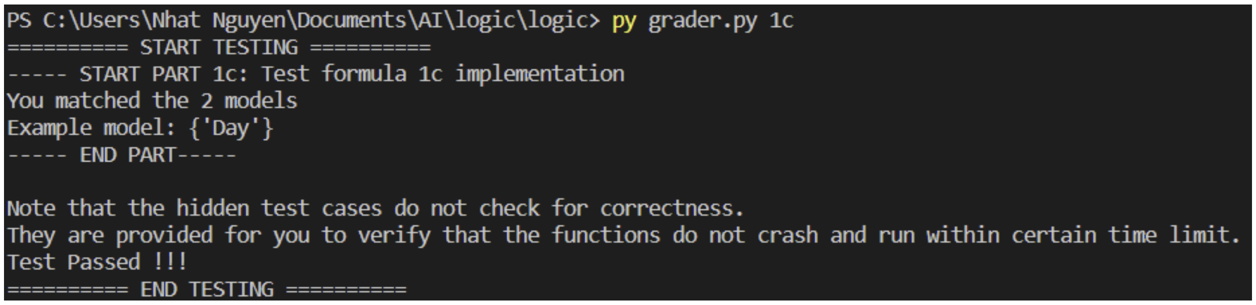
```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 1b
========== START TESTING ==========
----- START PART 1b: Test formula 1b implementation
You matched the 4 models
Example model: {'Rain', 'Wet'}
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

Code:



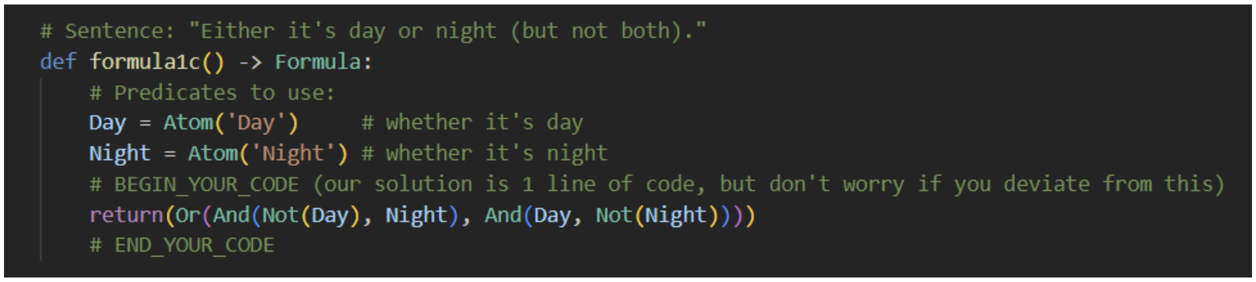
```python
# Sentence: "It's wet if and only if it is raining or the sprinklers are on."
def formula1b() -> Formula:
    # Predicates to use:
    Rain = Atom('Rain')                # whether it is raining
    Wet = Atom('Wet')                  # whether it it wet
    Sprinklers = Atom('Sprinklers')  # whether the sprinklers are on
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    return Equiv(Wet, Or(Rain, Sprinklers))
    # END_YOUR_CODE
```

1c



```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 1c
========== START TESTING ==========
----- START PART 1c: Test formula 1c implementation
You matched the 2 models
Example model: {'Day'}
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

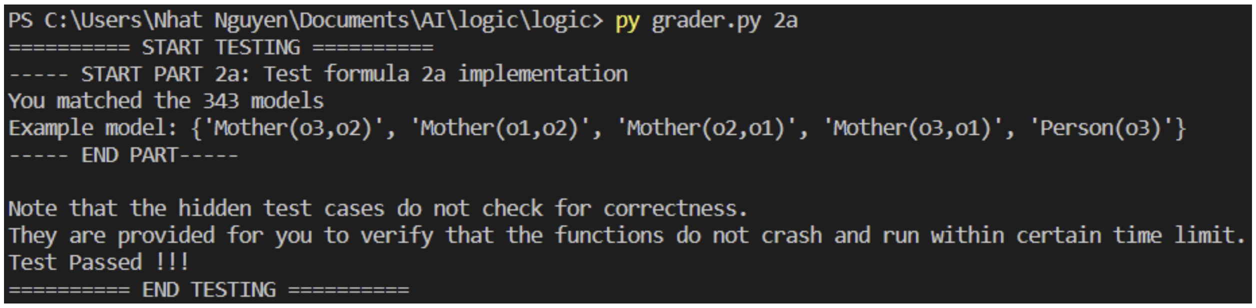
Code:



```python
# Sentence: "Either it's day or night (but not both)."
def formula1c() -> Formula:
    # Predicates to use:
    Day = Atom('Day')      # whether it's day
    Night = Atom('Night') # whether it's night
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    return(Or(And(Not(Day), Night), And(Day, Not(Night))))
    # END_YOUR_CODE
```

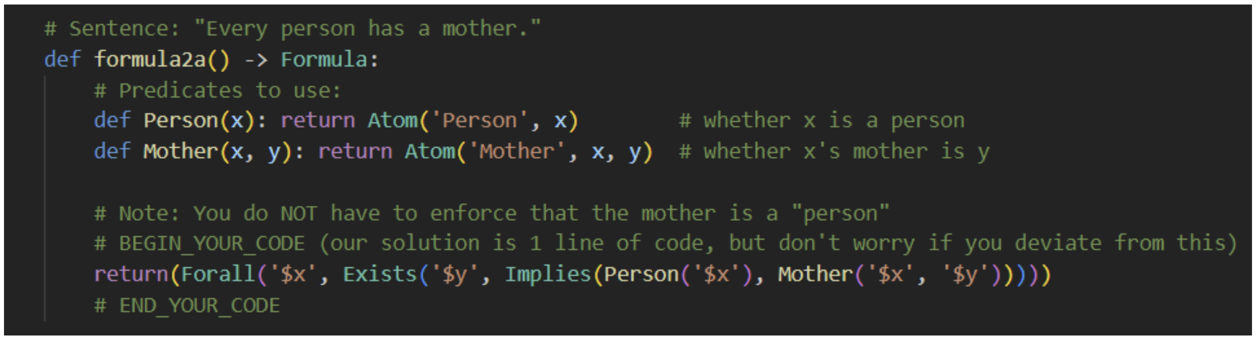
Part 2

2a



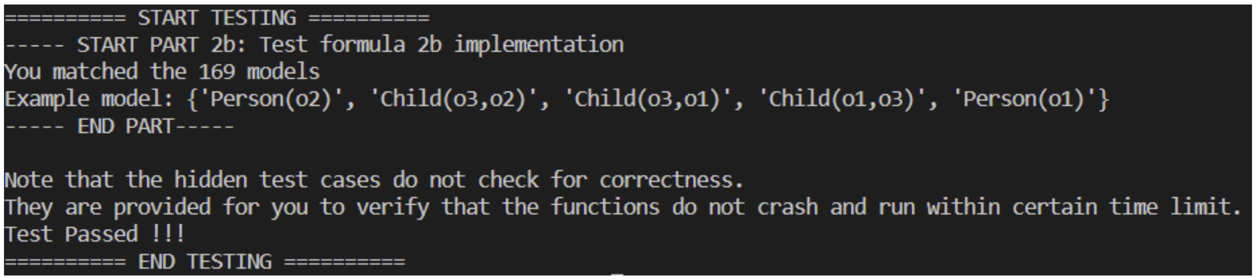
```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 2a
========== START TESTING ==========
----- START PART 2a: Test formula 2a implementation
You matched the 343 models
Example model: {'Mother(o3,o2)', 'Mother(o1,o2)', 'Mother(o2,o1)', 'Mother(o3,o1)', 'Person(o3)'}
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

Code:



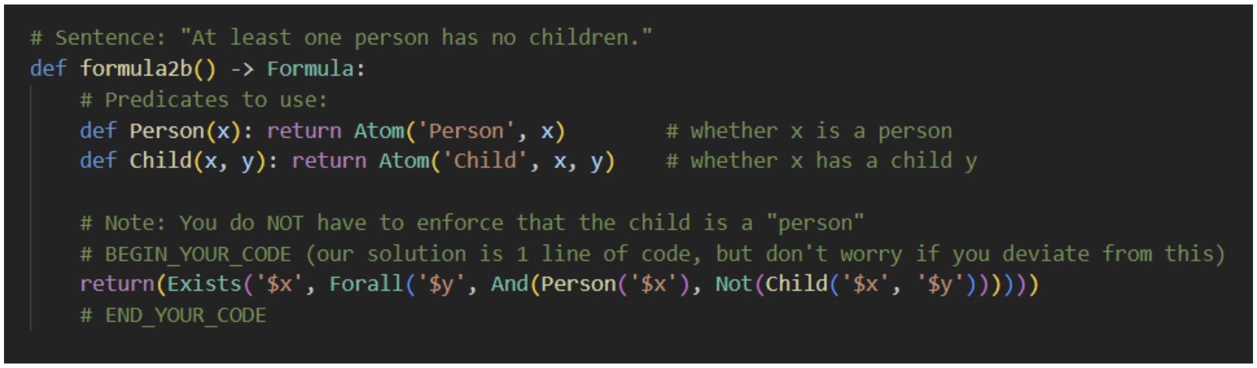
```python
# Sentence: "Every person has a mother."
def formula2a() -> Formula:
    # Predicates to use:
    def Person(x): return Atom('Person', x)        # whether x is a person
    def Mother(x, y): return Atom('Mother', x, y)  # whether x's mother is y

    # Note: You do NOT have to enforce that the mother is a "person"
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    return(Forall('$x', Exists('$y', Implies(Person('$x'), Mother('$x', '$y')))))
    # END_YOUR_CODE
```

2b



```
========== START TESTING ==========
----- START PART 2b: Test formula 2b implementation
You matched the 169 models
Example model: {'Person(o2)', 'Child(o3,o2)', 'Child(o3,o1)', 'Child(o1,o3)', 'Person(o1)'}
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

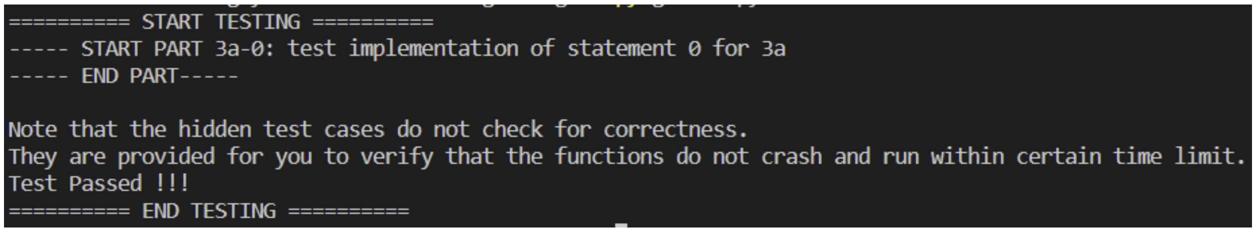
Code:



```python
# Sentence: "At least one person has no children."
def formula2b() -> Formula:
    # Predicates to use:
    def Person(x): return Atom('Person', x)        # whether x is a person
    def Child(x, y): return Atom('Child', x, y)    # whether x has a child y

    # Note: You do NOT have to enforce that the child is a "person"
    # BEGIN_YOUR_CODE (our solution is 1 line of code, but don't worry if you deviate from this)
    return(Exists('$x', Forall('$y', And(Person('$x'), Not(Child('$x', '$y'))))))
    # END_YOUR_CODE
```

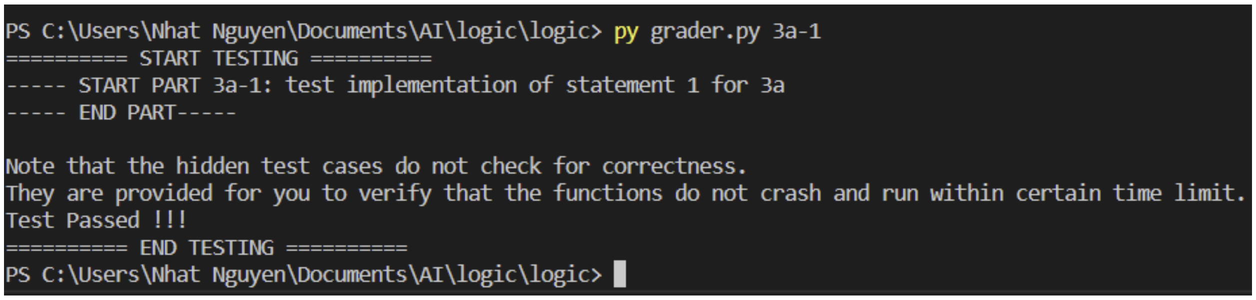
Part 3

3a-0



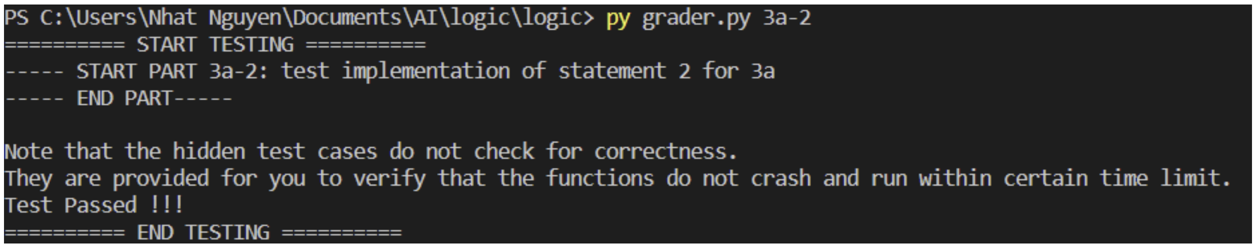
```
========== START TESTING ==========
----- START PART 3a-0: test implementation of statement 0 for 3a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

3a-1



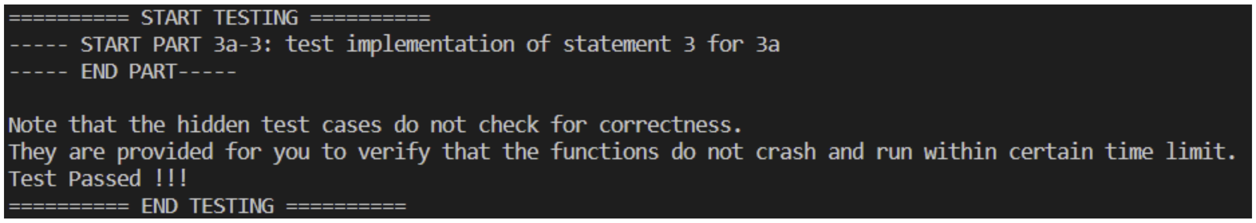
```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 3a-1
========== START TESTING ==========
----- START PART 3a-1: test implementation of statement 1 for 3a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic>
```

3a-2



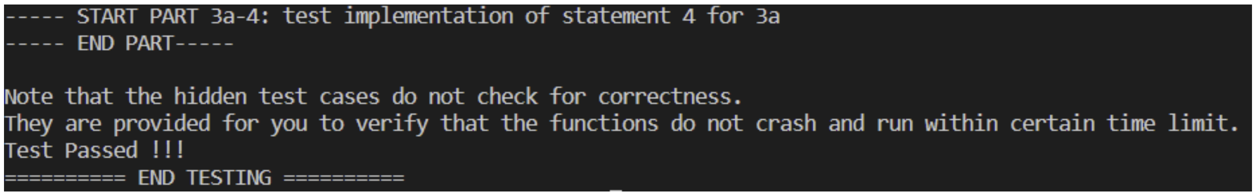
```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 3a-2
========== START TESTING ==========
----- START PART 3a-2: test implementation of statement 2 for 3a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

3a-3



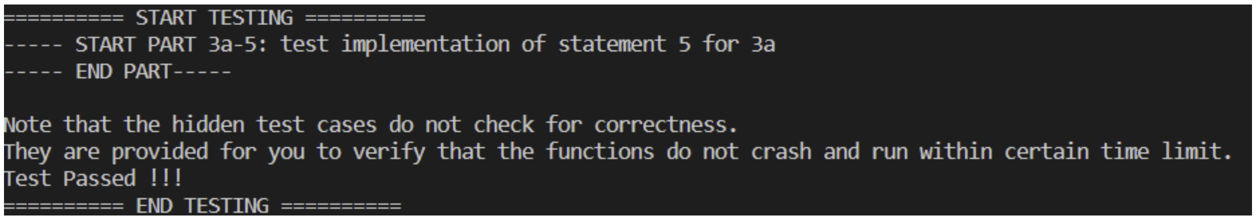
```
========== START TESTING ==========
----- START PART 3a-3: test implementation of statement 3 for 3a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

3a-4



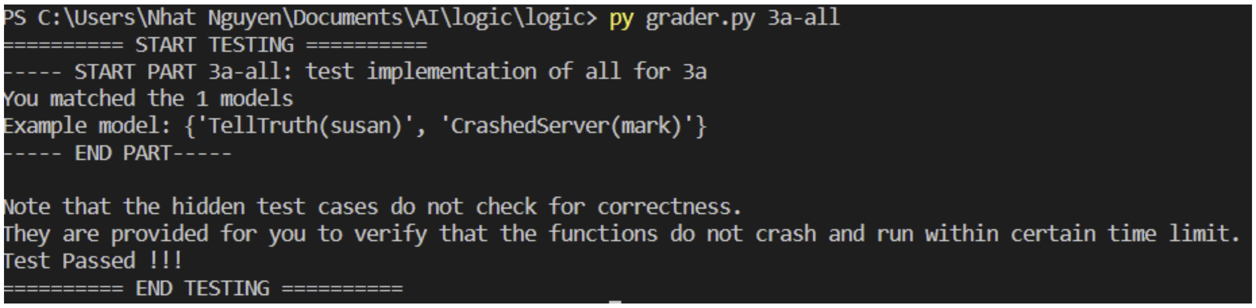
```
----- START PART 3a-4: test implementation of statement 4 for 3a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

3a-5



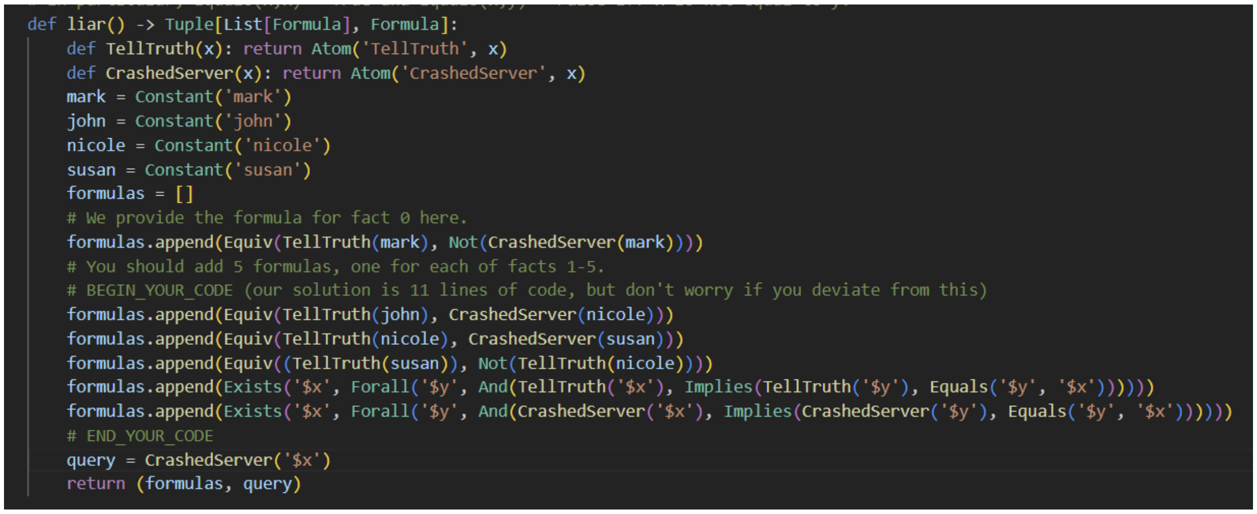
```
========== START TESTING ==========
----- START PART 3a-5: test implementation of statement 5 for 3a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

3a-all



```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 3a-all
========== START TESTING ==========
----- START PART 3a-all: test implementation of all for 3a
You matched the 1 models
Example model: {'TellTruth(susan)', 'CrashedServer(mark)'}
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

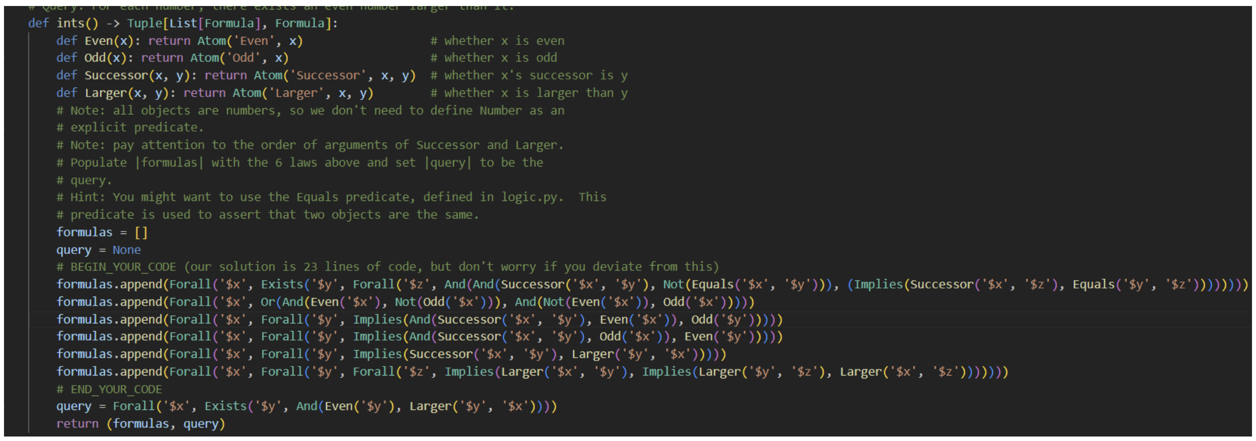
3a Code:



```python
def liar() -> Tuple[List[Formula], Formula]:
    def TellTruth(x): return Atom('TellTruth', x)
    def CrashedServer(x): return Atom('CrashedServer', x)
    mark = Constant('mark')
    john = Constant('john')
    nicole = Constant('nicole')
    susan = Constant('susan')
    formulas = []
    # We provide the formula for fact 0 here.
    formulas.append(Equiv(TellTruth(mark), Not(CrashedServer(mark))))
    # You should add 5 formulas, one for each of facts 1-5.
    # BEGIN_YOUR_CODE (our solution is 11 lines of code, but don't worry if you deviate from this)
    formulas.append(Equiv(TellTruth(john), CrashedServer(nicole)))
    formulas.append(Equiv(TellTruth(nicole), CrashedServer(susan)))
    formulas.append(Equiv((TellTruth(susan)), Not(TellTruth(nicole))))
    formulas.append(Exists('$x', Forall('$y', And(TellTruth('$x'), Implies(TellTruth('$y'), Equals('$y', '$x'))))))
    formulas.append(Exists('$x', Forall('$y', And(CrashedServer('$x'), Implies(CrashedServer('$y'), Equals('$y', '$x'))))))
    # END_YOUR_CODE
    query = CrashedServer('$x')
    return (formulas, query)
```

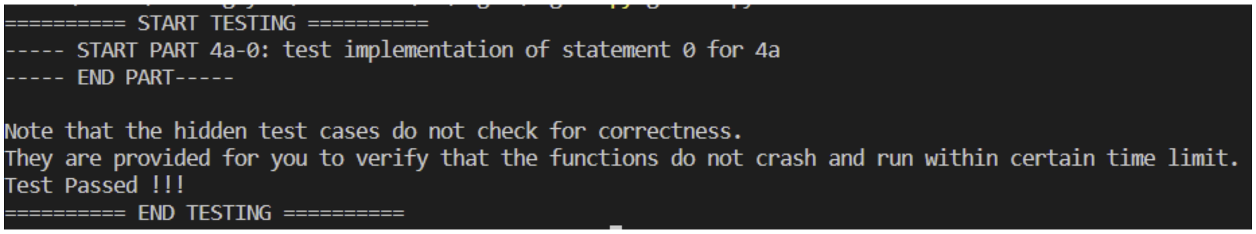
Part 4

4a-Code:



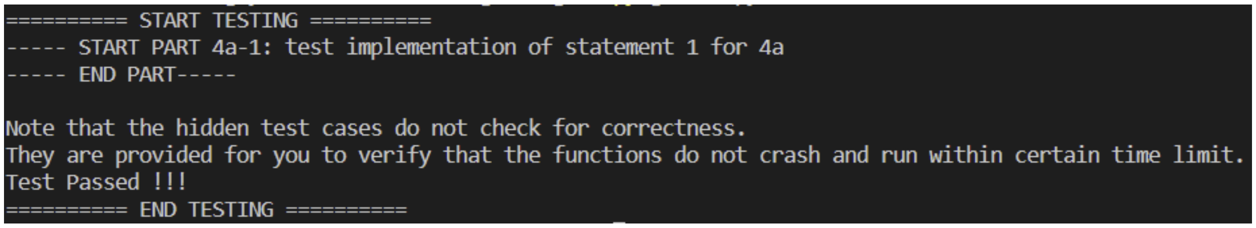
```
    # Query: For each number, there exists an even number larger than it.
    def ints() -> Tuple[List[Formula], Formula]:
        def Even(x): return Atom('Even', x)                   # whether x is even
        def Odd(x): return Atom('Odd', x)                     # whether x is odd
        def Successor(x, y): return Atom('Successor', x, y)  # whether x's successor is y
        def Larger(x, y): return Atom('Larger', x, y)        # whether x is larger than y
        # Note: all objects are numbers, so we don't need to define Number as an
        # explicit predicate.
        # Note: pay attention to the order of arguments of Successor and Larger.
        # Populate |formulas| with the 6 laws above and set |query| to be the
        # query.
        # Hint: You might want to use the Equals predicate, defined in logic.py.  This
        # predicate is used to assert that two objects are the same.
        formulas = []
        query = None
        # BEGIN_YOUR_CODE (our solution is 23 lines of code, but don't worry if you deviate from this)
        formulas.append(Forall('$x', Exists('$y', Forall('$z', And(And(Successor('$x', '$y'), Not(Equals('$x', '$y'))), (Implies(Successor('$x', '$z'), Equals('$y', '$z'))))))))
        formulas.append(Forall('$x', Or(And(Even('$x'), Not(Odd('$x'))), And(Not(Even('$x')), Odd('$x')))))
        formulas.append(Forall('$x', Forall('$y', Implies(And(Successor('$x', '$y'), Even('$x')), Odd('$y')))))
        formulas.append(Forall('$x', Forall('$y', Implies(And(Successor('$x', '$y'), Odd('$x')), Even('$y')))))
        formulas.append(Forall('$x', Forall('$y', Implies(Successor('$x', '$y'), Larger('$y', '$x')))))
        formulas.append(Forall('$x', Forall('$y', Forall('$z', Implies(Larger('$x', '$y'), Implies(Larger('$y', '$z'), Larger('$x', '$z')))))))
        # END_YOUR_CODE
        query = Forall('$x', Exists('$y', And(Even('$y'), Larger('$y', '$x'))))
        return (formulas, query)
```

4a-0



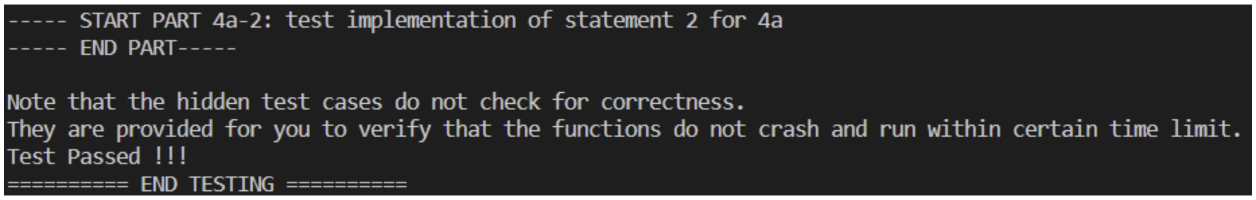
```
========== START TESTING ==========
----- START PART 4a-0: test implementation of statement 0 for 4a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

4a-1



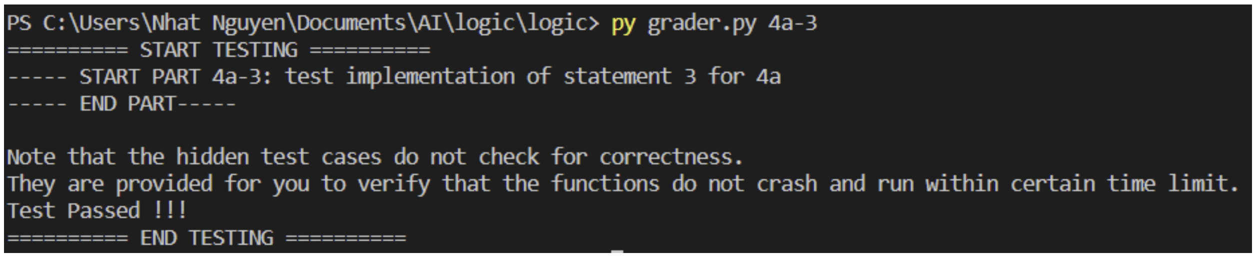
```
========== START TESTING ==========
----- START PART 4a-1: test implementation of statement 1 for 4a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

4a-2



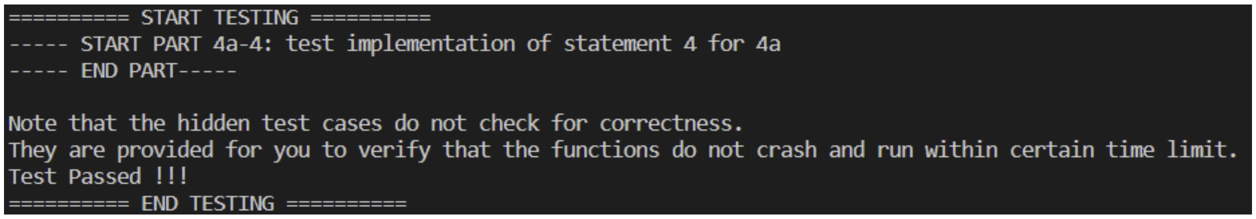
```
----- START PART 4a-2: test implementation of statement 2 for 4a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

4a-3



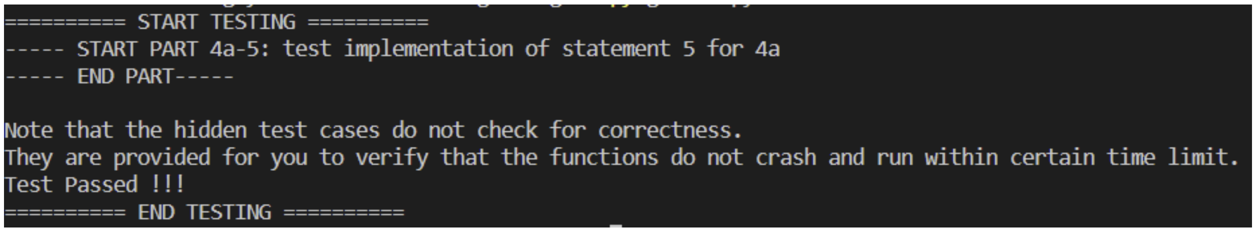
```
PS C:\Users\Nhat Nguyen\Documents\AI\logic\logic> py grader.py 4a-3
========== START TESTING ==========
----- START PART 4a-3: test implementation of statement 3 for 4a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

4a-4



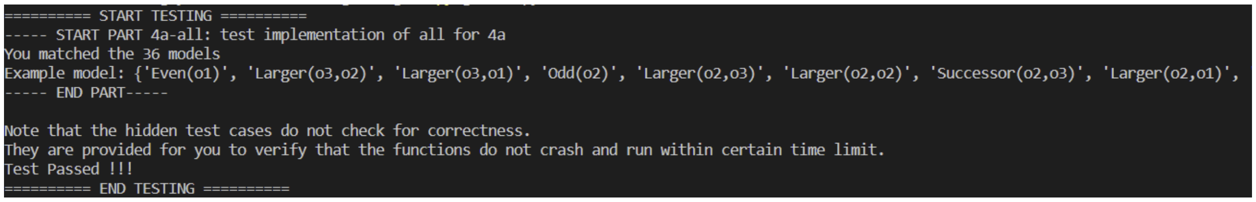
```
========== START TESTING ==========
----- START PART 4a-4: test implementation of statement 4 for 4a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

4a-5



```
========== START TESTING ==========
----- START PART 4a-5: test implementation of statement 5 for 4a
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```

4a-all



```
========== START TESTING ==========
----- START PART 4a-all: test implementation of all for 4a
You matched the 36 models
Example model: {'Even(o1)', 'Larger(o3,o2)', 'Larger(o3,o1)', 'Odd(o2)', 'Larger(o2,o3)', 'Larger(o2,o2)', 'Successor(o2,o3)', 'Larger(o2,o1)',
----- END PART-----

Note that the hidden test cases do not check for correctness.
They are provided for you to verify that the functions do not crash and run within certain time limit.
Test Passed !!!
========== END TESTING ==========
```