```python
'''
Name: Chongyi Xu
Student ID: 1531273
Course: Math 381, Fall 2017
Title: HW3 Python Scripts and Outputs
Instructor: Dr. Matthew Conroy
Due: Friday, October 20, 2017

Doing the first question
Define a graph G = (V, E) as follows.
Let V = {1, 2, 3, . . . , 10}.
Define E = {(i, j) : i, j \in V, i \neq j, i + 4j is prime or j + 4i is prime}.
Create and solve (using lpsolve) an IP to find the chromatic number of G, χ(G).
'''
from lpsolve55 import * # Import LP solve

NUM = 10 # Intialize the 'n' value

def is_prime(a):
    '''
    Decide if a number is prime

    Parameters
    ---------
    a: the number want to decide
    '''
    return all(a % i for i in range(2, a))


def findEdges(num):
    '''
    Find all edges

    Parameters
    ----------
    num: the value of 'n'

    Returns
    ---------
    edges: a set of edges
    '''
    edges = []
    for i in range(num):
        for j in range(num):
            if i != j and (is_prime(i + 4*j) or is_prime(4*i + j)):
                edges.append([i, j]) # Add edges to the edge set
    return edges

edge_set = findEdges(NUM) # find the edge set

def nums(num, count):
    '''
    Generate a list of given number for given count of times

    Parameters
    --------
    num: desired value
    count: times that the number repeats
    '''
    result = []
    for i in range(count):
        result.append(num)
    return result

# Generate a lpsolver
```

```
lp = lpsolve('make_lp', 0, 110)
for i in range(110):
    # constraint (4)
    ret = lpsolve('set_binary', lp, i, True) # Set all variables to be binary

x_coe = nums(0, 100)
y_coe = nums(1, 10)
obj_coe = x_coe + y_coe
# Set up the objective function
# Minimize y1 + ... + yn
ret = lpsolve('set_obj_fn', lp, obj_coe)

# sum x_{ik} from k = 1 to n, for i = 1, ..., n
# Loop over i and k to set up the constraint for the sum of x_{ik}
constraint_1 = nums(0, 110)
for i in range(NUM):
    for k in range(NUM):
        constraint_1[i * 10 + k] = 1 # set the coefficients of x_ik
    ret = lpsolve('add_constraint', lp, constraint_1, EQ, 1)
    constraint_1 = nums(0, 110) # reset the coefficients

# x_{ik} <= y_{k}  <=> x_{ik} - y_{k} <= 0 where i, k = 1, ..., n
# Loop over i to set up the constraint for every single x_{ik}
constraint_2 = nums(0, 110)
for i in range(NUM):
    for k in range(NUM):
        constraint_2[i * 10 + k] = 1 # set the coefficient of x_ik
        constraint_2[100 + k] = -1 # set the coefficient of y_k
        ret = lpsolve('add_constraint', lp, constraint_2, LE, 0)
        constraint_2 = nums(0, 110) # reset the coefficients

# x_{ik} + x_{jk} <= 1 for all edges in edge set for k = 1, ... ,n
# Use for loops for i, j to detect if (vi, vj) is in the edge set
constraint_3 = nums(0, 110)
for i in range(NUM):
    for j in range(NUM):
        for k in range(NUM):
            if [i, j] in edge_set:
                constraint_3[i * 10 + k] = 1 # set the coefficient of x_ik
                constraint_3[j * 10 + k] = 1 # set the coefficient of x_jk
                ret = lpsolve('add_constraint', lp, constraint_3, LE, 1)
                constraint_3 = nums(0, 110) # reset the coefficients


lpsolve("solve", lp)
print(lpsolve('get_objective', lp))
print(lpsolve('get_variables', lp)[0])


''' output --------------------------------------------------------
[Running] python "c:\Users\Johnnia\Desktop\46\Fall 2017\Math381\HW3\tempCodeRunnerFile.py"
set_binary: Column 0 out of range

Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:      550 constraints,     110 variables,         1180 non-zeros.
Sets:                                   0 GUB,                  0 SOS.

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.


Relaxed solution                   1 after         96 iter is B&B base.
```

```

Feasible solution                  4 after        156 iter,         4 nodes (gap 150.0%)

Optimal solution                   4 after      11939 iter,      2636 nodes (gap 150.0%).

Relative numeric accuracy ||*|| = 1.11022e-16

 MEMO: lp_solve version 5.5.2.5 for 64 bit OS, with 64 bit REAL variables.
      In the total iteration count 11939, 112 (0.9%) were bound flips.
      There were 1319 refactorizations, 0 triggered by time and 1 by density.
       ... on average 9.0 major pivots per refactorization.
      The largest [LUSOL v2.2.1.0] fact(B) had 1544 NZ entries, 1.0x largest basis.
      The maximum B&B level was 33, 0.2x MIP order, 5 at the optimal solution.
      The constraint matrix inf-norm is 1, with a dynamic range of 1.
      Time to load data was 0.009 seconds, presolve used 0.000 seconds,
       ... 0.526 seconds in simplex solver, in total 0.535 seconds.
4.0
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0]
[Done] exited with code=0 in 1.175 seconds

-------Conclusion-------------------------------------------------------------
The last 10 columns of the coefficiet of variables [1.0, 1.0, 1.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0] tells that y1, y2, y3, y9 are selected. In the other
word, 4 is the chromatic number for this graph G(V, E) (4 color is needed for
coloring this graph), which is as same as the output of calling "get_objective"
through lpsolve. One interesting fact is that, the color is not actually selected
"one-by-one" (from color 1 to 2) but jumped from color 3 directly to color 9.
'''

```