Olga Ibáñez Solé
NetID: oibanez
N° de matricule : 000426526
INFO-101: Groupe 1

```python
def negli_diff(a, b):
    """
    Takes two numbers a and b and returns True if the difference (diff) between them is
    negligible and False if it is not. A value is considered to be negligibe  if
    |a-b|/(|a|+|b|) < 1e-5
    """
    diff = abs(a-b)/(abs(a) + abs(b))
    if diff < 1e-5:
        return True
    else:
        return False

def lcm(a, b):
    """
    Takes two integers a and b and returns their least common multiple (lcm), calculated
    as a*b/gcd (greatest common divisor).
    """
    a, b = abs(a), abs(b)
    a, b = max(a, b), min(a, b)
    x, y = a, b
    while y > 0:
        x, y = y, x%y
    return (a*b)//x


def compute_combili(eq1, eq2, var_elim):
    """
    Takes two equations eq1 and eq2 and the name of a variable(var_elim) and returns an
    equation which is a linear combination of eq1 and eq2 where var_elim has been
    eliminated (by reduction).
    """

    multiple = lcm(eq1[var_elim], eq2[var_elim])

    c1 = multiple//eq1[var_elim]
    c2 = multiple//eq2[var_elim]

    eq1 = {var:eq1[var]*c1 for var in eq1}
    eq2 = {var:eq2[var]*c2 for var in eq2}

    combili = {var:eq1[var]-eq2[var] for var in eq1}

    return combili
```

Olga Ibáñez Solé
NetID: oibanez
N° de matricule : 000426526
INFO-101: Groupe 1

```
"""
The function negli_diff() does not apply here since the compute_combili() works with
integer coefficients, but could be useful when working with float numbers.
"""

def find_var(equation):
    """
    Takes an equation and returns a variable whose coefficient is not zero (if it exists,
    if it does not, it returns None).
    """
    for var in equation:
        if equation[var] != 0 and var != 'ti':
            return var
    return None

def triangulation(system):
    """
    Takes a system of n equations and returns an equivalent triangular system (if
    possible). Otherwise, it returns None.
    """
    for i in range(len(system)):
        x = find_var(system[i])
        if x == None:
            return None
        for j in range(i+1, len(system)):
            if system[j][x] != 0:
                system[j] = compute_combili(system[i], system[j], x)
    return system

def substitution(system):
    """
    Takes a triangular system of equations and returns the solution of the system in the
    form of a dictionary whose keys are the names of the variables and whose values are
    the solutions.
    """
    for i in reversed(range(len(system))):
        x = find_var(system[i])
        x_val = -system[i]['ti']/system[i][x]

        for j in reversed(range(i)):
                        system[j]['ti'] += x_val*system[j][x]
                        system[j][x] = 0
    result = {find_var(system[i]):-system[i]['ti']/system[i][find_var(system[i])] for i
in range(len(system))}
    return result
```

NetID: oibanez
N° de matricule : 000426526
INFO-101: Groupe 1

```
def print_equation(equation, order):
    """
    Takes an equation (in the form of a dictionary) and a list of the variables present
    in the equation in the order they were typed by the user in the execution of
    encode_system(). prints the equation in human readable form.
    """
    eq_string = ''
    for var in order:
        if equation[var] != 0:
            if equation[var] > 0:
                eq_string += ' + '
            else:
                eq_string += ' - '
            if abs(equation[var]) != 1 or var == 'ti':
                eq_string += str(abs(equation[var]))
            if var != 'ti':
                eq_string += var
    eq_string += ' = 0'
    eq_string = eq_string.strip(' + ')
    print(eq_string)

def print_system(system, order):
    """
    Takes a system and a list of the variables present in the equation in the order they
    were typed by the user in the execution of encode_system(). prints the system in
    human readable form.
    """
    for equation in system:
        print_equation(equation, order)


def encode_system():
    """
    Prompts the user to type the names and the coefficients of the variables of a system
    of equations. Checks that the number of variables and the number of equations are
    equal as well as the validity of the data introduced and prints each equation
    separately and the whole system at the end. Returns the linear system of equations in
    the form of a dictionary.
    """
# Encode variables' names
    ok = False
    while not ok:
        variables = input("Encoder les inconnues (lettres entre a et z séparées "
                          "par un espace):")
        var_list = variables.split(' ')
```

Olga Ibáñez Solé

NetID: oibanez
N° de matricule : 000426526
INFO-101: Groupe 1

```
    ok = True
    for var in var_list:
        if len(var)!=1 or  var < 'a' or var > 'z' or var_list.count(var) > 1:
            ok = False
            print("Erreur: toutes les inconnues doivent être des caractères "
                  "entre 'a' et 'z' et n'apparaître qu'une fois)")
            break

N = len(var_list)

question = 'Coefficients de'
for var in var_list:
    question += ' ' + var
question += " et du terme indépendent à l'équation n° "
var_list.append('ti')
system = []

# Encode variables' coefficients and print each of the equations
for i in range(N):

    ok = False
    while not ok:
        coefficients = input(question + str(i+1) + ': ')
        coef_list = coefficients.split(' ')
        ok = True

        try:
            message = ''
            if len(coef_list) != N+1:
                message = 'Erreur: on attendait ' + str(N+1) + ' coefficients'
                raise ValueError

            coef_list = [int(coef) for coef in coef_list]
            d = dict(zip(var_list, coef_list))
            system.append(d)

        except ValueError:
            if message == '':
                message = 'Tous les coefficients doivent être des nombres entiers'
            print(message)
            ok = False

    print('\nEquation encodée:')
    print_equation(d, var_list)
```

Olga Ibáñez Solé
NetID: oibanez
N° de matricule : 000426526
INFO-101: Groupe 1

```
    # Display and return the whole system
    print('\nSystème:')
    print_system(system, var_list)

    return system

def solve_system():
    """
    Calls encode_system() so that the user can type the names of the variables and the
    coefficients of each equation and passes the system of equations (list of
    dictionaries) as an argument to the function triangulation(). if the system can be
    solved, it applies the function substitution() to the triangular system and returns
    the solution(in the form of a dictionary). It it is indetermined or impossible, it
    prints an Error message.
    """
    system = encode_system()
    systriangulaire = triangulation(system)
    if systriangulaire == None:
        print("\nLe système est indeterminé ou impossible")
        return None
    else:
        solution = substitution(systriangulaire)
        print('Solution: ')
        for var in solution:
            print(var, '=', solution[var])

if __name__ == "__main__":

    solve_system()
```