```python
"""
    Akshay Sharma
    16123004
    Maths And Computing
    Date of submission - 9 April 2018

    Numerical Techniques
    Assignment 2
"""


def factorial(n):
    if n > 1:
        return n * factorial(n - 1)
    else:
        return 1


# Evaluates p*(p-1)*(p-2).....(p-n)
def expression_p(p, n):
    if n > 1:
        return (p - n + 1) * expression_p(p, n - 1)
    else:
        return p


# Evaluates q*(q+1)*(q+2).....(q+n)
def expression_q(q, n):
    if n > 1:
        return (q + n - 1) * expression_q(q, n - 1)
    else:
        return q


# Make them global, for frequent access
X = []
Y = []
diff_table = []   # The difference table


def initialize(values):
    """
    Constructs X, Y and diff_table
    :param values: List of tuples (x[i],y[i])
    """

    # Constructing Lists X and Y
    for v in values:
        X.append(v[0])
        Y.append(v[1])

    diff_table.extend([[0 for a in X] for b in Y])

    # Adding the first values
    for i in range(len(X)):
        diff_table[i][0] = Y[i]

    # Constructing the diff table
    for i in range(1, len(X)):
        for j in range(len(X) - i):
            diff_table[j][i] = diff_table[j + 1][i - 1] - diff_table[j][i - 1]

    # Uncomment to view the difference table
    pretty_print(diff_table)


def newtons_forward_interpolation(x):
    """
```

```
    :param x: At what 'x' to interpolate
    :return: f(x) at that point
    """

    p = (x - X[0]) / (X[1] - X[0])
    result = Y[0]

    for i in range(1, len(X)):
        result += expression_p(p, i) * diff_table[0][i] / factorial(i)

    return result


def newtons_backward_interpolation(x):
    """
    :param x: At what 'x' to interpolate
    :return: f(x) at that point
    """

    q = (x - X[len(X) - 1]) / (X[2] - X[1])
    result = Y[len(X) - 1]

    for i in range(1, len(X)):
        result += expression_q(q, i) * diff_table[len(X) - i - 1][i] / factorial(i)

    return result


def lagrange_interpolation(x):
    """
        :param x: At what 'x' to interpolate
        :return: f(x) at that point
    """

    sum_of_terms = 0
    for i in range(len(X)):
        temp = Y[i]
        for j in range(len(X)):
            if j != i:
                temp *= (x - X[j]) / (X[i] - X[j])
        sum_of_terms += temp

    return sum_of_terms


# Util Function for printing a matrix neatly
def pretty_print(matrix):
    s = [[str(e) for e in row] for row in matrix]
    lens = [max(map(len, col)) for col in zip(*s)]
    fmt = '\t'.join('{{:{}}}'.format(x) for x in lens)
    table = [fmt.format(*row) for row in s]
    print 'The difference table is:'
    print '\n'.join(table)


initialize([(1, 0), (2, 7.01), (3, 25.91), (4, 63.014), (5, 123.909)])
print newtons_backward_interpolation(3.5)

'''
    Interpolating Polynomial Values

    The difference table is:
    0           7.01    11.89   6.314   -0.727
    7.01        18.9    18.204  5.587   0
    25.91       37.104  23.791  0       0
    63.014      60.895  0       0       0
    123.909     0       0       0       0
```

```
    Command: initialize([(1, 0), (2, 7.01), (3, 25.91), (4, 63.014), (5, 123.909)])
             print newtons_forward_interpolation(3.5)
    Output:  41.8202734375


    Command: initialize([(1, 0), (2, 7.01), (3, 25.91), (4, 63.014), (5, 123.909)])
             print newtons_backward_interpolation(4.5)
    Output:  90.1668359375


    Command: initialize([(1, 0), (2, 7.01), (3, 25.91), (4, 63.014), (5, 123.909)])
             print lagrange_interpolation(3.5)
    Output:  41.8202734375
'''
```