Experiment No.: 08
Experiment Name: Implementation of Newton's Divide Difference Interpolation.
Theory:
Newton's Divided Difference Interpolation is a numerical method used to estimate or find the value of a function for a given point when the actual function is not known but a set of data points is available.
It is especially useful when the data points are unequally spaced.
Suppose you are given n+1 data points:

$$(x0, y0), (x1, y1), (x2, y2), \dots, (xn, yn)$$

where Type equation here.
Newton's divided difference formula gives the interpolating polynomial as:

$$P(x) = f[x0] + (x - x0)f[x0, x1] + (x - x0)(x - x1)f[x0, x1, x2] + \cdots + (x - x0)(x - x1) \dots (x - xn - 1)f[x0, x1, \dots, xn]$$

Here,

- $f[xi] = first\ value\ of\ the\ function$
- $f[xi, xi + 1] = \frac{f[xi+1] - f[xi]}{xi+1-xi}$
- $f[xi, xi + 1, xi + 2] = \frac{f[xi+1, xi+2] - f[xi, xi+1]}{xi+1-xi}$, and so on.

These are called divided differences, and they are calculated step by step in a difference table.

Program 1: Programming Code

```
1. import pandas as pd
2.
3. def calculate_x_minus_xi(i, val, x):
4.     prod = 1
5.     for j in range(i):
6.         prod *= (val - x[j])
7.     return prod
8.
9. def newton_divided_difference(x, y, val):
10.          n = len(x)
11.
12.          table = [[None] * n for _ in range(n)]
13.
14.          for i in range(n):
15.              table[i][0] = y[i]
16.
17.          for j in range(1, n):
18.              for i in range(n - j):
19.                  table[i][j] = (table[i+1][j - 1] - table[i][j - 1]) /
    (x[i+j] - x[i])
20.
21.          col_names = [f"Δ^{i}y" for i in range(n)]
```

```
        df = pd.DataFrame(table, columns=col_names)
        df.insert(0, "x", x)

        print("=> By using Newton Divided Difference Interpolation:\n")
        print(df.round(4).fillna(""))

        result = table[0][0]
        for i in range(1, n):
            result += table[0][i] * calculate_x_minus_xi(i, val, x)

        print(f"\nApproximate result at point {val} is: {result:.5f}")
        return None


    x = [3.6, 3.8, 3.9, 4.0, 4.2]
    y = [1.675, 1.436, 1.318, 1.246, 1.128]
    val = 4.1

    newton_divided_difference(x, y, val)
```

Output:

```
Active code page: 65001

D:\GitHub002\04 Fourth Semester\CSE 2206_Numerical Methods Lab>
al Methods Lab\lab9\NewtonsDivideDifferanceInterpolation.py"
=> By using Newton Divided Difference Interpolation:

     x   Δ^0y   Δ^1y    Δ^2y    Δ^3y     Δ^4y
0  3.6  1.675 -1.195    0.05   5.625 -17.1528
1  3.8  1.436  -1.18     2.3 -4.6667
2  3.9  1.318  -0.72  0.4333
3  4.0  1.246  -0.59
4  4.2  1.128

Approximate result at point 4.1 is: 1.20229

D:\GitHub002\04 Fourth Semester\CSE 2206_Numerical Methods Lab>
```

Discussion & Conclusion

Newton's Divided Difference method builds the interpolation polynomial incrementally.
Each divided difference represents the contribution of higher-order terms in the polynomial.
The structure of the formula makes it efficient and flexible—especially when new data points are added. It also forms the basis of Newton's Forward and Backward Interpolation when the data points are equally spaced.However, the accuracy of the interpolation depends on the number of points used and how well they represent the actual function. Using too many points might cause oscillations in the interpolated curve.