Practical – 8
Newton Interpolation

```
newtonDividedDifference[x_List, y_List] :=
 Module[{n = Length[x], dd, i, j}, dd = Table[0, {n}, {n}];
  Do[dd[[i, 1]] = y[[i]], {i, 1, n}];
  For[j = 2, j ≤ n, j++, For[i = j, i ≤ n, i++,
     dd[[i, j]] = (dd[[i, j - 1]] - dd[[i - 1, j - 1]]) / (x[[i]] - x[[i - j + 1]]);];];
  dd]


newtonPolynomial[x_List, y_List, var_Symbol] :=
 Module[{dd = newtonDividedDifference[x, y], n = Length[x], poly}, poly = dd[[1, 1]];
  Do[poly = poly + dd[[i, i]] * Product[var - x[[k]], {k, 1, i - 1}], {i, 2, n}];
  Expand[poly]]


xVals = {0.5, 1.5, 3, 5, 6.5, 8};
yVals = {1.625, 5.875, 31, 131, 282.125, 521};
P = newtonPolynomial[xVals, yVals, x]
f7 = P /. x → 7
```

Out[30]= $1. + 1.\,x + 1.\,x^3$

Out[31]= 351.

```
In[44]:= NDD[x0_, f0_, startindex_, endindex_] :=
  Module[{x = x0, f = f0, i = startindex, j = endindex, answer},
   If[i == j, Return[f[[i]]], answer =
     (NDD[x, f, i + 1, j] - NDD[x, f, i, j - 1]) / (x[[j]] - x[[i]]);
    Return[answer]];];
x = {0.5, 1.5, 3, 5, 6.5, 8};
f = {1.625, 5.875, 31, 131, 282.125, 521};
NDD[x, f, 1, 2]
```

Out[47]= 4.25

```
In[48]:= NDDP[x0_, f0_] :=
           Module[{x1 = x0, f = f0, n, newtonPolynomial, k, j},
            n = Length[x1];
            newtonPolynomial[y_] = 0;
            For[i = 1, i ≤ n , i++, prod[y_] = 1;
             For[k = 1, k ≤ i - 1, k++, prod[y_] = prod[y] * (y - x1[[k]])];
             newtonPolynomial[y_] =
              newtonPolynomial[y] + NDD[x1, f, 1, i] * prod[y]];
            Return[newtonPolynomial[y]];];
         nodes = {0, 1, 3};
         values = {1, 3, 55};
         NDDP[nodes, values]

Out[51]= 1 + 2 y + 8 (-1 + y) y
```