1. Write a program based on computing divided differences one diagonal at a time as shown below:

\[
\begin{bmatrix}
  f(x_0) \\
  f(x_1) \\
  \vdots \\
  f(x_i) \\
\end{bmatrix} \rightarrow \begin{bmatrix}
  f(x_0) & f(x_0, x_1) \\
  f(x_1) & f(x_1, x_2) \\
  \vdots & \vdots \\
  f(x_i) & f(x_i, x_{i+1}) \\
\end{bmatrix} \rightarrow \ldots
\]

You will need nested loops for this. Define the MATLAB function as follows:

```matlab
function D = divdiff2(x, y)
```

Then use `divdiff2` in place of `divdiff` and see whether you get the same answer.

See the MATLAB function `divdiff2.m`

We produce, as in `divdiff.m`, an upper-triangular array `D`

\[
\begin{bmatrix}
  f(x_0) & f(x_0, x_1) & \cdots & f(x_0, x_1, \ldots, x_{m-1}) & f(x_0, x_1, \ldots, x_m) \\
  f(x_1) & f(x_1, x_2) & \cdots & f(x_1, x_2, \ldots, x_m) \\
  \vdots & \vdots & & \vdots & \vdots \\
  f(x_{i-1}) & f(x_{i-1}, x_i) \\
  f(x_i) & \vdots & & \vdots & \vdots \\
  \vdots & \vdots & & \vdots & \vdots \\
  f(x_{m-1}) & f(x_{m-1}, x_m) \\
  f(x_m) & & & & \\
\end{bmatrix}
\]

We write one loop `i = 1:m` going down the first column and then for each `i` we insert `D(i,1) = y(i)` and compute the diagonal for `j = 2:i`, which represents the elements `D(i,1), D(i-1,2), D(i-2,3), \ldots, D(1,i)` , that is, the elements `D(i-j+1,j)`.

2. Write a MATLAB function

```matlab
function y = cubeinterp(x, xdata, ydata)
```

to do cubic tabular interpolation as discussed in the online example: You are given arrays `xdata` and `ydata`, where the `xdata` is ordered, i.e. 

\[xdata(1) < xdata(2) < \ldots < xdata(n)\]

Given a value of `x`, determine an index `k` for which 

\[xdata(k) <= x <= xdata(k+1)\]

Then interpolate the four points 

\[(x_j, y_j), j = k-1, k, k+1, k+2\]

with a cubic polynomial `p_3` and output `y = p_3(x)` for the given `x`.

Note that if `x < x(2)` or `x > x(n-1)` the cubic interpolation cannot be performed, and an error message should be output. Also, you may assume that `x` is at most a one-dimensional array, in which case the output `y` should be the same size as `x` (you will need to create a loop and do the calculation for each element of `x` individually).
We present two versions of this, \( y = \text{cubeinterp}(x, x_{\text{data}}, y_{\text{data}}) \) and 
\( y = \text{cubeinterp2}(x, x_{\text{data}}, y_{\text{data}}) \). The first conforms to the problem as stated; 
in the second we allow \( x \) to be any value and use the nearest 4 datapoints to 
do the interpolation. Each function consists of two basic steps: first, 
given the value of \( x \), find the four datapoints that define the cubic 
polynomial; second we do the interpolation using divided differences and 
then evaluate the resulting Newton-form polynomial at \( x \).

When you have a working function, investigate as follows the error estimate for this 
method \( |e(h)| \leq \frac{3}{128} \max_{x_{k-1} \leq t \leq x_{k+2}} |f^{(4)}(t)| h^4 \) using as an example data from \( f(x) = \sin(\pi x/2) \) 
on \([0, 1]\) with data spaced by \( h = .1 \) and \( h = .05 \) (include one datapoint to the left of 0 
and one to the right of 1 so that the entire interval \([0, 1]\) can be covered): calculate and 
plot the error for values of \( x \) in \([0, 1]\) and compare with the plot of \( \frac{3}{128} f^{(4)}(x) h^4 \).

Comment on what you see.

```matlab
>> xdata=-.1:.1:1.1;xdata=xdata';ydata=f(xdata);
>> x=0:.01:1;y=cubeinterp(x,xdata,ydata);
>> plot(x,y,x,f(x),xdata,ydata,'o')
```

The interpolant is visually identical to the original function when \( h = .1 \). We 
now plot the error for \( h = .1 \) and \( h = .2 \), along with the plot of \( \frac{3}{128} f^{(4)}(x) h^4 \)
```matlab
>> plot(x,f(x)-y,x,3*(.1)^4*(pi/2)^4*f(x)/128)
>> h=.05;
>> xdata=-h:h:1+h;xdata=xdata';ydata=f(xdata);
>> x=0:.005:1;y=cubeinterp(x,xdata,ydata);
>> plot(x,f(x)-y,x,3*h^4*(pi/2)^4*f(x)/128);
```
We observe that the error estimate is very accurate midway between the datapoints. Obviously, at the datapoints the error is zero because we interpolated there - however midway between the datapoints, the maximum value of \((x-x_{k-1}) \cdot (x-x_{k+2})\) is achieved and the value of \(f^{(4)}(\xi)\) is very close to \(f^{(4)}(x)\) because \(\xi\) is in a short range of \(x_{k-1} < x < x_{k+2}\).

3. Interpolate the following functions on the stated interval \([a, b]\) at equally spaced interpolation nodes of spacing \(\frac{b-a}{n}\) where \(n = 5, 10, 15\), in each case plotting the interpolant, the original function, and the datapoints. Make a separate plot of the error \(f(x) - p(x)\). Comment as appropriate on what you see. You don’t need to write your own interpolation functions - use the ones we developed in class or modify them if you like.

Assuming \(f, a, b\), are given we create an M-file `hw6prob2.m` with the statements

```matlab
for n=5:5:15
    x=a:(b-a)/n:b;y=f(x);
    x=x';y=y';
    c=polyint(x,y); %interpolates and graphs interpolant
    t=a:(b-a)/200:b;s=f(t);
    hold on; plot(t,s,'r');
    pause; hold off
    y=newtpoly(t,c,x); %calculates values of interpolant
    plot(t,s-y); %gives the error plot
    pause
end
```

and then from the command window we work our way through the cases:

```matlab
» f=inline('sin(x)');a=-pi;b=pi; » hw6prob3
» f=inline('abs(x)');a=-1;b=1; » hw6prob3
» f=inline('1./(1+x.^2)');a=-5;b=5; » hw6prob3
```
a) $f(x) = \sin(x)$, on $[-\pi, \pi]$

The error converges rapidly to zero as $n$ increases.
b) $f(x) = |x|$ on $[-1, 1]$

The error is still significant at $n = 15$ and may (actually does) grow with increasing $n$. 
c) $f(x) = \frac{1}{1 + x^2}$ on $[-5, 5]$

The error is not decreasing, and will actually get much worse if $n$ is increased further.
We are considering \( f(x) = \ln x \) on \([1, 2]\) with 10 nodes equally spaced. Thus in our usual notation we have degree \( n = 9 \) and \( h = 1/9 \). We will assemble the inequalities we need in terms of a general value of \( n \) and then put \( n = 9 \).

The error is given by \( e(x) = \frac{1}{(n + 1)!} f^{(n+1)}(\xi)(x - x_0) \cdot \cdots (x - x_n) \). Using the inequality in the textbook, \(|(x - x_0) \cdots (x - x_n)| \leq \frac{n!}{4} h^{n+1}\). Now \( f^{(n+1)}(x) = (-1)^n \frac{n!}{x^{n+1}} \), so for \( 1 < \xi < 2 \) we have \( |f^{(n+1)}(\xi)| = \left| (-1)^n \frac{n!}{\xi^{n+1}} \right| \leq n! \). Putting together the inequalities, we have

\[
|e(x)| \leq \frac{1}{(n + 1)!} n! \frac{n!}{4} h^{n+1} = \frac{1}{4(n + 1)} n! h^{n+1} = \frac{1}{40} 9! \left( \frac{1}{9} \right)^{10} = 2.6018 \times 10^{-6}
\]