Exercise 1: Problem 10.5

We are defining

$$< a|b >= \int_{-1}^{1} ab(1-x^2)^{-1/2}dx$$

Find the orthonormal basis using $[1, x, x^2]$. Starting with 1 lets normalize note $< 1|1 >= \int_{-1}^{1}(1-x^2)^{-1/2}dx = \sin^{-1}(x)|_{-1}^{1} = \pi$. Thus $q_0 = 1/\sqrt{\pi}$. Note $< x|1/\sqrt{\pi} >= 1/\sqrt{\pi}\int_{-1}^{1} x(1-x^2)^{-1/2}dx = -1/\sqrt{\pi}\sqrt{1-x^2}|_{-1}^{1} = 0$ and $< x|x >= \int_{-1}^{1} x^2(1-x^2)^{-1/2}dx = 1/2(-x\sqrt{1-x^2} + \sin^{-1}(x)) = \pi/2$, thus $q_1 = x\sqrt{2/pi}$. Note $1/\sqrt{\pi} < x^2|1/\sqrt{\pi} >= 1/\pi\int_{-1}^{1} x^2(1-x^2)^{-1/2}dx = 1/2$, and $< x^2|x\sqrt{2/pi} >= \sqrt{2/pi}\int_{-1}^{1} x^3(1-x^2)^{-1/2}dx = 0$ and $< x^2 - 1/2|x^2 - 1/2 >= \pi/8$ (last integral via WRA), thus $q_2 = x^2\sqrt{8/\pi} - \sqrt{2/\pi}$.
Note that $q_1 = \alpha\cos(0)$, $q_1 = \alpha\cos(\arccos(x))$, $q_2 = \alpha(2x^2+1) = \cos(2\arccos(x))$ where the various αs are constants so our basis is the same as the one given in the book.

Exercise 2: Problem 10.7

```
function mintegral=traprule(f,a,b,nm)
    mintegral=[];
    for(n=nm)
        w=(b-a)./n;
        cent=linspace(a+w,b-w,n-1);
        integral=w.*(1/2*f(a)+sum(f(cent))+1/2*f(b));
        mintegral=[mintegral integral];
    end
endfunction
```

```
function mintegral=simpson(f,a,b,nm)
    mintegral=[];
    for(n=nm)
        w=(b-a)./n;
        #there are n+1 points each with a midpoint thus n midpoints
        points=f(linspace(a,b,n+1));
        centers=f(linspace(a+w/2,b-w/2,n));
        estint=w/6*(points(1:n)+4*centers+points(2:n+1));
        integral=sum(estint);
        mintegral=[mintegral integral];
    end
endfunction
```

Note that realInt came from the described quad evaluation.

```
>> f=@(x) cos(x.^2);
>> n=10.^pow;
>> h=1./n;
>> errh=abs(traprule(f,0,1,n)-realInt);
```

```
>> [h',errh',(errh./(h.^2))']
ans =

   1.0000e-01   1.4025e-03   1.4025e-01
   1.0000e-02   1.4025e-05   1.4025e-01
   1.0000e-03   1.4025e-07   1.4025e-01
   1.0000e-04   1.4025e-09   1.4025e-01
   1.0000e-05   1.4027e-11   1.4027e-01
   1.0000e-06   1.2335e-13   1.2335e-01

>> n=3.^pow;
>> h=1./n
>> errh=abs(simpson(f,0,1,n)-realInt);
>> [h',errh',(errh./(h.^4))']
ans =

   3.3333e-01   1.3129e-06   1.0635e-04
   1.1111e-01   9.9952e-09   6.5579e-05
   3.7037e-02   1.5784e-10   8.3883e-05
   1.2346e-02   1.9957e-12   8.5910e-05
   4.1152e-03   2.3981e-14   8.3616e-05
   1.3717e-03   2.2204e-16   6.2712e-05

>> diary off
```

Exercise 3: Continuing with the theme that some sample points are better than others, recall that polynomial interpolation with high-order polynomials is prone to making large oscillation errors, but that this can be minimized using Chebyshev polynomials, which are the Lagrange polynomials associated with the sample points $x_j = cos(\pi + (\pi j/n))$, $j = 0, \cdots, n$ on the interval $[-1, 1]$. Clenshaw-Curtis integration is integration using polynomial interpolation at these sample points. Use the MATLAB polyfit function to perform polynomial interpolation at these sample points for $n = 4, 6, 10$ and then use the resulting polynomials to approximate

$$\int_{-1}^{1} x\sin(x)dx$$

Compare your approximations to the exact answer (which you should compute by hand). Integration by parts!

integration by parts (tabular method)

$$\begin{array}{ccc} + & x & \sin(x) \\ - & 1 & -\cos(x) \\ + & 0 & -\sin(x) \end{array}$$

Thus

$$\int_{-1}^{1} x\sin(x)dx = 2(\sin(1) - \cos(1))$$

so our error for the three would be

```
realInt=2*(sin(1)-cos(1));
for i=4:2:8
    ceb=cos(pi+pi*[0:i]/i);
    f=@(x) x.*sin(x);
    p=polyfit(ceb,f(ceb),i);
    pow=[i+1:-1:1];
    xp=(1-(-1).^pow)./pow;
    inte=xp*p';
    err=abs(inte-realInt)
end
```

```
>> cheb
err =    1.5436e-04
err =    1.4867e-07
err =    1.9205e-10
>> diary off
```

Exercise 4: Recall that 5 point Gauss-Legendre integration uses sample points$[-\beta, -\alpha, 0, \alpha, \beta]$ where

$$\alpha = 1/3\sqrt{5 - 2\sqrt{10/7}}$$

$$\beta = 1/3\sqrt{5 + 2\sqrt{10/7}}$$

Write a code that performs composite Gauss-Legendre integration with these sample points. Your code should have the signature
q=glquad(f,a,b,N)
were f is the function to integrate, a and b are the endpoints of integration, and N is the number of subintervals. Your code should perform Gauss-Legendre integration on each subinterval and add them up. Then apply your function to compute

$$\int_{-1}^{1} x\sin(x)dx$$

using $N = 1, 2, 4, 10$. Compare the results of Gauss-Legendre integration to the results you saw using Clenshaw-Curtis integration.

```
function integrate=glquad(f,a,b,N)
lin=linspace(a,b,N+1);
suma=0;
for i=1:N
    suma+=recurse(f,lin(i),lin(i+1));
end
integrate=suma;



```

```

endfunction


function inte=recurse(f,a,b)
alp=1./3*sqrt(5-2\sqrt(10/7));
bet=1./3*sqrt(5+2\sqrt(10/7));
gl=([-bet,-alp,0,alp,bet]+1)/2*(b-a)+a;


p=polyfit(gl,f(gl),4);
pow=[5:-1:1];
xp=((b).^pow-(a).^pow)./pow;
inte=xp*p';


endfunction
```

```
realInt=2*(sin(1)-cos(1));
for i=[1, 2, 4,10]
    f=@(x) x.*sin(x);
    err=abs(glquad(f,-1,1,i)-realInt)
end
```

```
>> glerr
err =    3.5271e-04
err =    4.7596e-06
err =    7.1806e-08
err =    2.9128e-10
>> diary off
>> diary off
```

Let's see where we can compare the two. When GL is using one interval it uses 5 points, so lets compare it to the 4 point lenshaw-Curtis, both are on the 10^{-4} error. When GL is using two intervals it uses 10 points, so lets compare it to the 8 point lenshaw-Curtis, GL has error 10^{-6}, while lenshaw-Curtis has error 10^{-10}. Clearly lenshaw-Curtis does better using the same number of sample points for this function.

Exercise 5: Problem 9.1

```
f=@(x) sin(x);
x=pi/6;
fpp=@(h) (f(x+h)-2*f(x)+f(x-h))./(h.^2);
exponent=[-1:-1:-16];
h=10.^exponent;
[h',fpp(h)',abs(fpp(h)+sin(x))',h.^2']
```

```
>> secderiv
ans =

   1.0000e-01  -4.9958e-01   4.1653e-04   1.0000e-02
   1.0000e-02  -5.0000e-01   4.1667e-06   1.0000e-04
   1.0000e-03  -5.0000e-01   4.1619e-08   1.0000e-06
   1.0000e-04  -5.0000e-01   3.0387e-09   1.0000e-08
   1.0000e-05  -5.0000e-01   5.9648e-07   1.0000e-10
   1.0000e-06  -4.9993e-01   6.6572e-05   1.0000e-12
   1.0000e-07  -4.9405e-01   5.9508e-03   1.0000e-14
   1.0000e-08  -1.1102e+00   6.1022e-01   1.0000e-16
   1.0000e-09   5.5511e+01   5.6011e+01   1.0000e-18
   1.0000e-10   0.0000e+00   5.0000e-01   1.0000e-20
   1.0000e-11   0.0000e+00   5.0000e-01   1.0000e-22
   1.0000e-12   5.5511e+07   5.5511e+07   1.0000e-24
   1.0000e-13   5.5511e+09   5.5511e+09   1.0000e-26
   1.0000e-14  -5.5511e+11   5.5511e+11   1.0000e-28
   1.0000e-15   0.0000e+00   5.0000e-01   1.0000e-30
   1.0000e-16  -5.5511e+15   5.5511e+15   1.0000e-32

>> diary("d3.txt");
```

If we notice that we are deviding by h^2 we can easily see why the error starts to build. Quite quickly h^2 becomes larger than machine precision an so it makes seance why our error gets big after that, to illustrate I have also given the table containing h^2 values.

Exercise 6: Problem 9.5

Recall via Taylor expansion $f(x+a) = f(x) + f'(x)a + f''(\xi)a^2/2$. So we note that $f(x+h) = f(x) + f'(x)h + f''(\xi_1)h^2/2$ where $\xi_1 \in (x, x+h)$. Note that $f(x+2h) = f(x) + f'(x)2h + f''(\xi_2)2h^2$ where $\xi_2 \in (x, x+2h)$. Thus

$$\frac{1}{2h}[-3f(x) + 4f(x+h) - f(x+2h)] =$$

$$\frac{1}{2h}[-3f(x) + 4(f(x) + f'(x)h + f''(\xi_1)h^2/2) - (f(x) + f'(x)2h + f''(\xi_2)2h^2)] =$$

$$\frac{1}{2h}[4f'(x)h + 4f''(\xi_1)h^2/2 - f'(x)2h - f''(\xi_2)2h^2] =$$

$$f'(x) + \frac{1}{2h}[4f''(\xi_1)h^2/2 - f''(\xi_2)2h^2] =$$

$$f'(x) + h[f''(\xi_1) - f''(\xi_2)]$$

Thus the error in this approximation is $h[f''(\xi_1) - f''(\xi_2)]$, this is interesting, if we have a function with a stable second derivative the error in this estimation will be tiny.