Problem 4.1: I am doing this quick and easy, i.e., I could do a procedure for \( T \) with \( N \) defined internal to the procedure.

\[
T := (f, h, N) \rightarrow \frac{1}{2} \left( f(a) + 2 \left( \sum_{j=1}^{N-1} f(a+jh) \right) + f(a+Nh) \right) h
\]

\[
(4*T(f,h/2,2*N)-T(f,h,N))/3;
\]

simplify(%);

\[
\frac{1}{3} \left( f(a) + 2f\left(a + \frac{1}{2} h\right) + f\left(a+h\right) \right) h - \frac{1}{6} \left( f(a) + f(a+h) \right) h
\]

\[
\frac{1}{6} hf(a) + \frac{2}{3} hf\left(a + \frac{1}{2} h\right) + \frac{1}{6} hf\left(a+h\right)
\]

Note this is Simpson's rule  nodes at \( a, a+h, a+2h \) with weights \( h/6, 4h/6, h/6 \) respectively

\[
(9*T(f,h/3,3*N)-T(f,h,N))/8;
\]

simplify(%);

\[
\frac{3}{16} \left( f(a) + 2f\left(a + \frac{1}{3} h\right) \right) + 2f\left(a + \frac{2}{3} h\right) + f\left(a+h\right) \right) h - \frac{1}{16} \left( f(a) + f(a+h) \right) h
\]

\[
\frac{1}{8} hf(a) + \frac{3}{8} hf\left(a + \frac{1}{3} h\right) + \frac{3}{8} hf\left(a + \frac{2}{3} h\right) + \frac{1}{8} hf\left(a+h\right)
\]

Note this is the Newton-Cotes rule for 4 nodes  \( a, a+h/3, a+2/3*h, a+h \) with weights \( h/8, 3h/8, 3h/8, h/8 \)

\[
(16*T(f,h/4,4*N)-T(f,h,N))/15;
\]

simplify(%);

\[
\frac{2}{15} \left( f(a) + 2f\left(a + \frac{1}{4} h\right) \right) + 2f\left(a + \frac{1}{2} h\right) + 2f\left(a + \frac{3}{4} h\right) + f\left(a+h\right) \right) h - \frac{1}{30} \left( f(a) + f(a+h) \right) h
\]

\[
\frac{2}{15} hf(a) + \frac{4}{15} hf\left(a + \frac{1}{4} h\right) + \frac{4}{15} hf\left(a + \frac{1}{2} h\right) + \frac{4}{15} hf\left(a + \frac{3}{4} h\right) + \frac{1}{10} hf\left(a+h\right)
\]

This is a good rule, but certainly not one of the Newton-Cotes rules

Problem 4.2 There are several ways to do this. By definition one can integrate and choose the constant of integration to get the integral to be zero: this is easy by hand. I will use the generating function (no

\[
\text{int}(1,x=0..1)
\]

\[
taylor(w*exp(x*w)/(exp(w)-1),w,5);
\]

\[
1 + \left( x - \frac{1}{2} \right) w + \left( \frac{1}{12} + \frac{1}{2} x^2 - \frac{1}{2} x \right) w^2 + \left( \frac{1}{6} x^3 + \frac{1}{12} x - \frac{1}{4} x^2 \right) w^3 + \left( -\frac{1}{720} + \frac{1}{24} x^4 + \frac{1}{24} x^2 \right) w^4 + O(w^5)
\]

Problem 4.3 Let's now do the whole procedure for finding the points and weights of for Gaussian integration with an arbitrarily prescribed weight, \( w(t) \) on \([a,b] \).

\[
\text{restart}:
\]
with(LinearAlgebra):
Digits := 30;
a := -1; b := 1; # b := Pi;
w := t -> 1; # sqrt(t); # sin(t)^2;
IP := proc(f, g) evalf(int(f(t)*g(t)*w(t), t=a..b)) end;

Note the inner product IP above

n := 3;

p := array(0..n);
p[0] := 1:
j := 'j'; k := 'k':
for j from 0 to n-1 do;
vv := t*p[j]:
for k from 0 to j do;
vv := vv-IP(t*p[j], p[k])/IP(p[k], p[k])*p[k]:
od:
p[j+1] := expand(vv):
od:
for j from 0 to n do p[j]; od;

Here we compute the roots of the nth polynomial. There are
fast solvers adapted to orthogonal polynomials --- if there is
time we will discuss in class when we get to numerical linear algebra.

Lagrange basis functions for points indexed by i from 1 to N

\[
L := \text{proc}(N::\text{integer}, i::\text{integer}, v::\text{Vector}, x)
\text{description "Lagrange function";}
\text{local j, T;}
\text{T := 1; for j from 1 to N do}
\text{if (j <> i) then T := T*(x-v[j])/(v[i]-v[j]) fi;}
\text{od; T;}
\text{end proc;}
\]

Computation of nodes x and weights

\[
\begin{align*}
\text{for N from 1 to n do}
tempX := \text{Vector}(N):
x := \text{Vector}(N):
\end{align*}
\]
\[
\text{y:=Vector(N): weights:=Vector(N): tempX:= fsolve(p[N],t): print("number of points" = N): print('nodes'): for j from 1 to N do x[j] :=tempX[j]: print(x[j]): od: print('weights'): for j from 1 to N do weights[j]:=int(L(N,j,x,t),t=a..b): print(weights[j]): od: od:
\]

"number of points" = 1 
\[
\text{nodes} \\
(-2.92649536567162872787048949180 \times 10^{-35})_1 \\
\text{weights} \\
2
\]

"number of points" = 2 
\[
\text{nodes} \\
-0.577350269189625764509148780503 \\
0.577350269189625764509148780503 \\
\text{weights} \\
0.999999999999999999999999999999 \\
0.999999999999999999999999999999 \\
\]

"number of points" = 3 
\[
\text{nodes} \\
-0.774596669241483377035853079956 \\
-1.23193160224957838757292675500 \times 10^{-35} \\
0.774596669241483377035853079956 \\
\text{weights} \\
0.555555555555555555555555555555 \\
0.888888888888888888888888888880 \\
0.555555555555555555555555555555 \\
\]

# Let's verify the three point rule for polynomials of degree $\leq 5$ (I went to 6 to show it is really exactly 5). 
> k:='k': 
  > j:='j': 
  > for k from 0 to 6 do A:=int(t^k,t=a..b): B:=0: 
  > for j from 1 to 3 do 
  > B:= B+weights[j]*x[j]^k: 
  > od: 
  > print(A-B): 
  > od: 
0. 
0. 
3. \times 10^{-30} 
0. 
2. \times 10^{-30} 
0.
Problem 4.4

```maple
> restart:

with(LinearAlgebra):
Digits := 30;
a := 0; b := 1;
w := t -> sqrt(t);
IP := proc( f, g ) evalf(int(f(t)*g(t)*w(t), t=a..b)) end;

Note the inner product IP above

n := 3;
p := array(0..n);
p[0] := 1:
j := 'j': k := 'k':
for j from 0 to n-1 do;
vv := t*p[j]:
for k from 0 to j do;
vv := vv - IP(t*p[j], p[k])/IP(p[k], p[k])*p[k]:
end:
p[j+1] := expand(vv):
end:
for j from 0 to n do p[j]; od;
```

Here we compute the roots of the nth polynomial. There are fast solvers adapted to orthogonal polynomials --- if there is time we will discuss in class when we get to numerical linear algebra.

Lagrange basis functions for points indexed by i from 1 to N

```maple
L := proc( N::integer, i::integer, v::Vector, x)
   description "Lagrange function";
   local j, T;
   T := 1:
   for j from 1 to N do
      if (j<>i) then T := T*(x-v[j])/(v[i]-v[j])
   fi;
   od;
   T;
end proc;
L := proc(N::integer, i::integer, v::Vector, x)
   local j, T;
```

(14)

(15)

(16)

0.045714285714285714285714285716
Computation of nodes x and weights

for N from 1 to n do
  tempX := Vector(N);
  x := Vector(N);
  y := Vector(N);
  weights := Vector(N);
  tempX := fsolve(p[N], t);
  print("number of points" = N):
  print('nodes'):
  for j from 1 to N do
    x[j] := tempX[j];
    print(x[j]):
  od:
  print('weights'):
  for j from 1 to N do
    weights[j] := int(L(N, j, x, t)*sqrt(t), t=a..b):
    print(weights[j]):
  od:
od:

"number of points" = 1
nodes
0.6000000000000000000000000000000

weights
2/3

"number of points" = 2
nodes
0.289949197925690302229151737848
0.8211619131854208881959373262
weights
0.27755998231061630134788531615
0.38911066843560536531878135053

"number of points" = 3
nodes
0.164710286896542421522796386004
0.549868499216443563908599462554
0.900805829271629399183988766832
weights
0.125782674328838847953649287589
0.307602367681912735506181300105
0.233281624655915083206836078975

Let's verify the three point rule for polynomials of degree <= 5.

> k := 'k':
  j := 'j':
  for k from 0 to 5 do A := int(t^k*sqrt(t), t=a..b):
    B := 0:
    for j from 1 to 3 do
      B := B + weights[j]*x[j]^k:
    od:
  od:
print(A-B):

od:

-2. 10^{-30}
-2. 10^{-30}
-1. 10^{-30}
-2. 10^{-30}
-3. 10^{-30}
-5. 10^{-30}