1. We wish to derive the composite Simpson's rule. Let $x_i = a + ih$, $h = \frac{b-a}{2}$ for $0 \le i \le n$. Then, we may break the integral as follows: $\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + ... + \int_{x_{n-1}}^{x_n} f(x)dx$. For each of these, we may approximate the integral with Simpson's rule, as follows:

$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + ... + \int_{x_{n-1}}^{x_n} f(x)dx \\ &\approx \frac{x_1 - x_0}{6}\left[f(x_0) + 4f\left(\frac{x_0+x_1}{2}\right) + f(x_1)\right] + \frac{x_2 - x_1}{6}\left[f(x_1) + 4f\left(\frac{x_1+x_2}{2}\right) + f(x_2)\right] + \\ &... + \frac{x_n - x_{n-1}}{6}\left[f(x_{n-1}) + 4f\left(\frac{x_{n-1}+x_n}{2}\right) + f(x_n)\right]\end{aligned}$$

Notice that, since we are dividing $[a, b]$ into evenly spaced subintervals, $x_i - x_{i-1} = \frac{x_n - x_0}{n} = h$ where n is the number of subintervals. Thus, we find that

$$\int_a^b f(x)dx \approx \frac{h}{6}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + 4\sum_{i=1}^{n} f\left(\frac{x_i + x_{i-1}}{2}\right) + f(x_n)\right]$$

is a composite Simpson's rule that will approximate for n subintervals. Now, let $[a, b] = [a, c] \cup [c, b]$ where $c = \frac{a+b}{2}$. Then, we find that

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^c f(x)dx + \int_c^b f(x)dx \\ &\approx \frac{b-a}{12}\left[f(a) + 2f(c) + 4f\left(\frac{a+c}{2}\right) + 4f\left(\frac{c+b}{2}\right) + f(b)\right].\end{aligned}$$

Now, let $n = 4$. Then, with $h = \frac{b-a}{4}$, we find that

$$\begin{aligned}\int_a^b f(x)dx &= \int_a^{a+h} f(x)dx + \int_{a+h}^{a+2h} f(x)dx + \int_{a+2h}^{a+3h} f(x)dx + \int_{a+3h}^{b} f(x)dx \\ &\approx \frac{b-a}{24}\left[f(a) + 2\sum_{i=1}^{3} f(a+ih) + 4\sum_{i=1}^{4} f\left(\frac{a+ih+a+(i-1)h}{2}\right) + f(b)\right].\end{aligned}$$

2. Here is my code:

```
---------------------------------------------------------------------------
type csimpson.m

function y = csimpson(f,a,b,n)
%y = csimpson(f,a,b,n)
%This is an algorithm by Alexander Winkles used to perform composite
%Simpson's rule to compute numerical integral values.
%
%f : the function being integrated
%a : the lower bound of the integral
%b : the upper bound of the integral
%n : the number of subintervals used

h=(b-a)/n;

x = zeros(1,n+1);

x(1) = a;
x(n+1) = b;

p = 0;
q = 0;

for i=2:n
```

```
    x(i) = a + h*(i-1);
end;

for i=2:n
    p = p + f(x(i));
end;

for i = 2:n+1
    q = q + f((x(i)+x(i-1))/2);
end;

y = (h/6)*(f(a) + 2*p + 4*q + f(b));

-------------------------------------------------------------------------

Here is my work. The expected value for the integral is 1/10, or 0.1000.

f =@(x) sin(10*x)

f =

    @(x)sin(10*x)

y = csimpson(f,0,pi/20,2)

y =

   0.100013458497419

0.1000 - y

ans =

    -1.345849741937744e-05

y = csimpson(f,0,pi/20,3)

y =

   0.100002631217059

0.1000 - y

ans =

    -2.631217059262392e-06

y = csimpson(f,0,pi/20,4)

y =

   0.100000829552397

0.1000 - y

ans =

    -8.295523967610796e-07

y = csimpson(f,0,pi/20,5)

y =

   0.100000339222090

0.1000 - y

ans =
```

```
    -3.392220900566567e-07

y = csimpson(f,0,pi/20,6)

y =

   0.100000163443858

0.1000 - y

ans =

    -1.634438579894981e-07

diary off
```

Thus, errors using this method are small, so the approximation is good.

3. Here is my code:

```
---------------------------------------------------------------------------
type romberg.m

function y = romberg(f,a,b,n,q1,q2)
%y = romberg(f,a,b,n,r,q)
%
%This is an algorithm written by Alexander Winkles that performs Romberg
%integration.
%
%f  : the function being integrated
%a  : the lower bound of the integral
%b  : the upper bound of the integral
%n  : the 2^n subinterval specification
%q1 : if q == 1, returns an array of all Romberg values computed
%q2 : a two element vector. if q2(1) == 1, then prints the error matrix
%       where q2(2) is the true value

h = b-a;

R = zeros(n+1,n+1);

R(1,1) = (1/2)*h*(feval(f,a) + feval(f,b));

for i = 2 : n+1
    h = h/2;
    sum = 0;
    for u = 1 : (2^(i-2))
        sum = sum + feval(f,(a + (2*u-1)*h));
    end;
    R(i,1) = (1/2)*R(i-1,1) + h*(sum);
    for j = 2 : i
        R(i,j) = R(i,j-1) + (R(i,j-1) - R(i-1,j-1))/(4^(j-1) - 1);
    end;
end;


if q1 == 1
    disp(R)
end;

if q2(1) == 1
    E = zeros(n+1,n+1);
    for i=1:n+1
        for j=1:n+1
            if i >= j
                E(i,j) = q2(2) - R(i,j);
            end;
        end;
```

```
    end;
    disp(E)
end;

y = R(n+1,n+1);
--------------------------------------------------------------------------
Here are my results, with the actual answer being (sin(5))^2/5:

f =@(x) sin(10*x)

f =

    @(x)sin(10*x)

romberg(f,0,1,5,1,[1 (sin(5)^2/5)])
  -0.272010555444685                   0                   0                   0                   0                   0
  -0.615467415053912  -0.729953034923654                   0                   0                   0                   0
   0.076384322692718   0.307001568608261   0.376131875510389                   0                   0                   0
   0.159313166073844   0.186956113867552   0.178953083551505   0.175823261456920                   0                   0
   0.177881250868329   0.184070612466491   0.183878245706420   0.183956422883482   0.183988317634175                   0
   0.182408071050210   0.183917011110837   0.183906771020460   0.183907223803223   0.183907030865653   0.183906951406446

   0.455917708352330                   0                   0                   0                   0                   0
   0.799374567961557   0.913860187831299                   0                   0                   0                   0
   0.107522830214927  -0.123094415700616  -0.192224722602744                   0                   0                   0
   0.024593986833801  -0.003048960959907   0.004954069356140   0.008083891450726                   0                   0
   0.006025902039316  -0.000163459558846   0.000028907201225  -0.000049269975837  -0.000081164726530                   0
   0.001499081857435  -0.000009858203192   0.000000381887185  -0.000000070895577   0.000000122041992   0.000000201501199


ans =

   0.183906951406446

diary off
```

Thus, errors using this method are small, so the approximation is good.

4. To create a Romberg formula based on the composite Simpson's rule, we simply substituted using our composite Simpson's rule for the usual trapezoid rule used to compute the $R(n, 0)$ values.

5. Here is my code:

```
--------------------------------------------------------------------------
type romberg2.m

function y = romberg2(f,a,b,n,q1,q2)
%y = romberg(f,a,b,n,r,q)
%
%This is an algorithm written by Alexander Winkles that performs Romberg
%integration from composite Simpson's rule.
%
%f  : the function being integrated
%a  : the lower bound of the integral
%b  : the upper bound of the integral
%n  : the 2^n subinterval specification
%q1 : if q == 1, returns an array of all Romberg values computed
%q2 : a two element vector. if q2(1) == 1, then prints the error matrix
%       where q2(2) is the true value

R = zeros(n+1,n+1);

for i=1:n+1
    R(i,1) = csimpson(f,a,b,n*i);
end;

for i=2:n+1
    for j=2:i
        R(i,j) = R(i,j-1) + (R(i,j-1) - R(i-1,j-1))/(4^(j-1) - 1);
```

```
    end;
end;

y = R(n+1,n+1);

if q1 == 1
    disp(R)
end;

if q2(1) == 1
    E = zeros(n+1,n+1);
    for i=1:n+1
        for j=1:n+1
            if i >= j
                E(i,j) = q2(2) - R(i,j);
            end;
        end;
    end;
    disp(E)
end;
--------------------------------------------------------------------------
Here are my results:

romberg2(f,0,1,5,1,[1 (sin(5)^2/5)])
   0.185064715255191                   0                   0                   0                   0                   0
   0.183972961249513   0.183609043247620                   0                   0                   0                   0
   0.183919935383224   0.183902260094461   0.183921807884250                   0                   0                   0
   0.183911173839661   0.183908253325140   0.183908652873852   0.183908444064163                   0                   0
   0.183908795455152   0.183908002660316   0.183907985949327   0.183907975363224   0.183907973525181                   0
   0.183907943875968   0.183907660016240   0.183907637173302   0.183907631637174   0.183907630289229   0.183907629953710

  -0.001157562347546                   0                   0                   0                   0                   0
  -0.000065808341867   0.000298109660025                   0                   0                   0                   0
  -0.000012782475579   0.000004892813184  -0.000014654976605                   0                   0                   0
  -0.000004020932016  -0.000001100417495  -0.000001499966206  -0.000001291156518                   0                   0
  -0.000001642547507  -0.000000849752670  -0.000000833041682  -0.000000822455579  -0.000000820617536                   0
  -0.000000790968323  -0.000000507108595  -0.000000484265656  -0.000000478729529  -0.000000477381584  -0.000000477046065


ans =

   0.183907629953710

diary off
```

Notice with this method that there is a larger error than with just the Romberg. This can be attributed to the error associated with each of the composite Simpson's errors adding up.