Problems 1 and 2

By now, it should be clear how I handle main.cpp, assign5.hpp, and in assign5.cpp void assign5::main(). Since these problems are small and simple enough, I kept all other code in assign5.cpp for this assignment.

Common code:

```
double Trapm(const std::vector<double> f, double h) {
    int n = f.size();
    double sum = f[0];
    for (int i = 1; i < n; ++i) {
        sum += 2 * f[i];
    }
    sum += f[n];
    return h * sum / 2;
}
double Trapm(const std::vector<double> x, const std::vector<double> y) {
    int n = x.size();
    double sum = 0;
    for (int i = 1; i < n; ++i) {
        sum += (y[i] + y[i - 1]) * (x[i] - x[i - 1]) / 2;
    }
    return sum;
}
double Trapm(std::function<double(double)> f, double a, double b, int n) {
    double h = (b - a) / n;
    double sum = f(a);
    for (int i = 1; i < n; ++i) {
        sum += 2 * f(a + h * i);
    }
    sum += f(b);
    return h * sum / 2;
}
```

Problem 1

21.9 Suppose that the upward force of air resistance on a falling object is proportional to the square of the velocity. For this case, the velocity can be computed as

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}\,t\right)$$

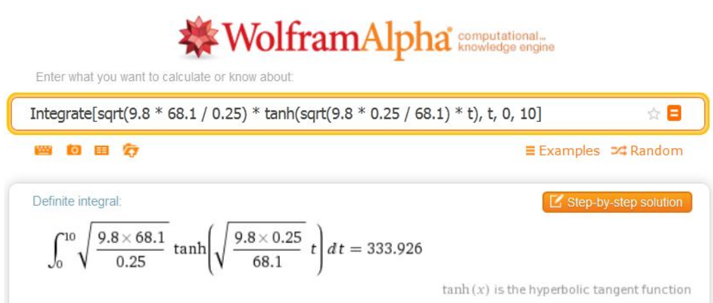
where c_d = a second-order drag coefficient. (a) If $g = 9.8$ m/s^2, $m = 68.1$ kg and $c_d = 0.25$ kg/m, use analytical integration to determine how far the object falls in 10 s. (b) Make the same evaluation, but evaluate the integral with the multiple-segment trapezoidal rule. Use a sufficiently high n that you get three significant digits of accuracy.

Part b

```
Problem 1:

n = 5    --> s = 330.922
n = 10   --> s = 333.179
n = 50   --> s = 333.896
n = 100  --> s = 333.919
n = 500  --> s = 333.926
n = 1000 --> s = 333.926
```

Part a: since we can use regular calculators I figured wolfram alpha would be fine.



Code for problem 1:

```
void Prob1() {
    const double a = 0, b = 10,
        g = 9.8,  m = 68.1, c = 0.25;

    auto v = [&g, &m, &c](double t) {
        return sqrt(g * m / c) *
            tanh(sqrt(g * c / m) * t);
    };

    std::cout
        << "Problem 1:" << std::endl << std::endl
        << "n = 5    --> s = " << Trapm(v, a, b, 5) << std::endl
        << "n = 10   --> s = " << Trapm(v, a, b, 10) << std::endl
        << "n = 50   --> s = " << Trapm(v, a, b, 50) << std::endl
        << "n = 100  --> s = " << Trapm(v, a, b, 100) << std::endl
        << "n = 500  --> s = " << Trapm(v, a, b, 500) << std::endl
        << "n = 1000 --> s = " << Trapm(v, a, b, 1000) << std::endl
        << std::endl;
}
```

Problem 2

PROBLEM [2] for Assignment 5: Using Program case (b), solve Problem 24.22:

24.22 A wind force distributed against the side of a skyscraper is measured as

Height, l, m	0	30	60	90	120	150	180	210	240
Force, $F(l)$, N/m	0	340	1200	1600	2700	3100	3200	3500	3800

Compute the net force and the line of action due to this distributed wind.

```
Problem 2: 526200
```

Code for problem 2:

```
void Prob2() {
    const std::vector<double> x{0, 30, 60, 90, 120, 150, 180, 210, 240};
    const std::vector<double> y{0, 340, 1200, 1600, 2700, 3100, 3200, 3500,
3800};
    std::cout << "Problem 2: " << Trapm(x, y) << std::endl << std::endl;
}
```

Problems 3 and 4

Common code:

```
double Trap(double h, double f0, double f1) {
    return h * (f1 + f0) / 2;
}
double Simp38(double h, double f0, double f1, double f2, double f3) {
    return 3 * h * (f0 + 3 * f1 + 3 * f2 + f3) / 8;
}
double Simp13m(double h, int n, const std::vector<double> y) {
    double sum = y[0];
    for (int i = 1; i < n - 2; i += 2) {
        sum += 4 * y[i] + 2 * y[i + 1];
    }
    sum += 4 * y[n - 1] + y[n];
    return h * sum / 3;
}
double SimpInt(const std::vector<double> y, double h) {
    int n = y.size() - 1;
    if (n == 1) {
        return Trap(h, y[1], y[0]);
    } else {
        double sum = 0; int m = n;
        if (n % 2 != 0) {// if we got here n > 1 is a given
            sum += Simp38(h, y[n - 3], y[n - 2], y[n - 1], y[n]);
            m -= 3;
        }
        if (m > 1) { sum += Simp13m(h, m, y); }
        return sum;
    }
}
```

Problem 3

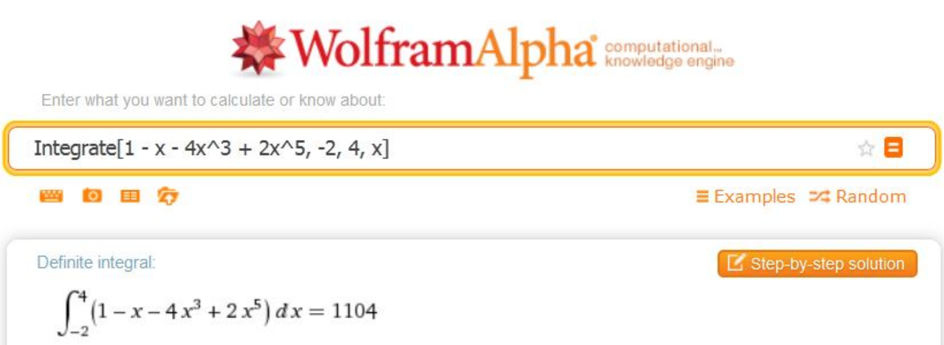
21.3 (modified) Evaluate the following integral

$$\int_{-2}^{4} \left(1 - x - 4 \cdot x^3 + 2 \cdot x^5\right) dx$$

reporting the following: (a) analytical solution (e.g., using Maxima). Then, calculate the numerical result using function SimpInt (b) for n = 2; (c) for n = 3; (c) for n = 4; (d) for n = 10; (e) for n = 20.

```
Problem 3:

n = 2  --> I = 1752
n = 3  --> I = 1392
n = 4  --> I = 1144.5
n = 10 --> I = 1105.04
n = 20 --> I = 1104.06
```



Code for problem 3:

```
void Prob3() {
    double a = -2, b = 4, h = 0;
    std::vector<double> y;
    auto f = [](double x) {
        double xpow[6];
        Powers(xpow, 6, x);
        return 1 - x - 4 * xpow[3] + 2 * xpow[5];
    };
    auto make = [&a, &b, &y, &f](int n, double& h) {
        y.resize(n + 1);
        h = (b - a) / n;
        y[0] = f(a);
        for (int i = 1; i < n; ++i) {
            y[i] = f(a + h * i);
        }
        y[n] = f(b);
    };

    std::cout << "Problem 3:" << std::endl << std::endl; make(2, h);
    std::cout << "n = 2  --> I = " << SimpInt(y, h) << std::endl; make(3, h);
    std::cout << "n = 3  --> I = " << SimpInt(y, h) << std::endl; make(4, h);
    std::cout << "n = 4  --> I = " << SimpInt(y, h) << std::endl; make(10,
h);
    std::cout << "n = 10 --> I = " << SimpInt(y, h) << std::endl; make(20,
h);
    std::cout << "n = 20 --> I = " << SimpInt(y, h) << std::endl <<
std::endl;
}
```

Problem 4:

PROBLEM [4] for Assignment 5: Using Program case (b), calculate the voltage for t = 1.2 s. (do not develop the plot of voltage versus time as suggested. Just calculate V(1.2).).

24.34 If a capacitor initially holds no charge, the voltage across it as a function of time can be computed as

$$V(t) = \frac{1}{C}\int_0^t i(t)\,dt$$

If $C = 10^{-5}$ farad, use the following current data to develop a plot of voltage versus time:

t, s	0	0.2	0.4	0.6	0.8	1	1.2
i, 10^{-3} A	0.2	0.3683	0.3819	0.2282	0.0486	0.0082	0.1441

```
Problem 4: 24.1593
```

Code for problem 4:

```
void Prob4() {
    const double h = 0.2;
    const std::vector<double> y{0.2, 0.3683, 0.3819, 0.2282, 0.0486, 0.0082,
0.1441};

    std::cout << "Problem 4: " <<  100 * SimpInt(y, h) << std::endl <<
std::endl;
}
```

Problems 5 and 6

Common code:

If you look at problems 1-2, the declaration of Trapm that accepts x and y is actually Trapun. I find it ridiculous to impose the assumption that data is equally spaced and I naturally programmed it to handle uneven data. No efficiency is lost this way, and it's just 100 times better, cleaner code.

PROBLEM [5] for Assignment 5: Using Program case (a), calculate the integral of problem 21.22 using the uneven trapezoidal rule.

21.22 The work produced by a constant temperature, pressure-volume thermodynamic process can be computed as

$$W = \int p\,dV$$

where W is work, p is pressure, and V is volume. Using a combination of the trapezoidal rule, Simpson's 1/3 rule, and Simpson's 3/8 rule, use the following data to compute the work in kJ (kJ = kN · m):

Pressure (kPa)	336	294.4	266.4	260.8	260.5	249.6	193.6	165.6
Volume (m^3)	0.5	2	3	4	6	8	10	11

```
Problem 5: 2671
```

Note: for the data shown, the volume, V, takes the role of x, while the pressure, p, becomes the y values.

Code for problem 5:

```
void Prob5() {
    const std::vector<double> v{0.5, 2, 3, 4, 6, 8, 10, 11};
    const std::vector<double> p{336, 294.4, 266.4, 260.8, 260.5, 249.6,
193.6, 165.6};
    std::cout << "Problem 5: " << Trapm(v, p) << std::endl << std::endl;
}
```

PROBLEM [6] for Assignment 5: Using Program case (a), calculate the integral of problem 24.4 using the uneven trapezoidal rule.

24.4 Integration provides a means to compute how much mass enters or leaves a reactor over a specified time period, as in

$$M = \int_{t_1}^{t_2} Qc\,dt$$

where t_1 and t_2 = the initial and final times, respectively. This formula makes intuitive sense if you recall the analogy between integration and summation. Thus, the integral represents the summation of the product of flow times concentration to give the total mass entering or leaving from t_1 to t_2. If the flow rate is constant, Q can be moved outside the integral:

$$M = Q\int_{t_1}^{t_2} c\,dt \qquad \text{(P24.4)}$$

Use numerical integration to evaluate this equation for the data listed below. Note that $Q = 4\ m^3/min$.

t, min	0	10	20	30	35	40	45	50
c, mg/m^3	10	35	55	52	40	37	32	34

```
Problem 6: 7880
```

Note: For the data shown, the time, t, takes the role of x in the integral, while the role of y is given by Q*c (not simply c).

Code for problem 6

```
void Prob6() {
    const std::vector<double> t{0, 10, 20, 30, 35, 40, 45, 50};
    const std::vector<double> c{10, 35, 55, 52, 40, 37, 32, 34};
    std::cout << "Problem 6: " << 4 * Trapm(t, c) << std::endl << std::endl;
}
```

Problem 7

What you need to do for Problem [7]
Use functions *TrapEq* and *Romberg*, as described above, to solve problems 22.2 and 22.3:

22.2 Use Romberg integration to evaluate

$$I = \int_1^2 \left(2x + \frac{3}{x}\right)^2 dx$$

to an accuracy of $\varepsilon_s = 0.5\%$ based on Eq. (22.9). Your results should be presented in the form of Fig. 22.3. Use the analytical solution of the integral to determine the percent relative error of the result obtained with Romberg integration. Check that ε_t is less than the stopping criterion ε_s.

22.3 Use Romberg integration to evaluate

$$\int_0^2 \frac{e^x \sin x}{1 + x^2} dx$$

to an accuracy of $\varepsilon_s = 0.5\%$. Your results should be presented in the form of Fig. 22.3.

RombergScript.sce

```
diary("Problems22_22_And_22_3.txt")

//Problem 22.2
a = 1; b = 2; maxiter = 50; ea = .5;

function [y] = f2(x)
    y = (2*x + (3 / x))^2;
endfunction;

[II, n1, iter1, ea1] = Romberg(a,b,maxiter,ea,f2)

//Problem 22.3
a = 0; b = 2; maxiter = 50; ea = .5;

function [z] = f3(t)
    z = (%e^(t) * sin(t)) / (1 + t^2);
endfunction;

[I2,n2,iter2,ea2] = Romberg(a,b,maxiter,ea,f3)

diary(0)
```

```
Output:
ans  =

   1.
 ea1  =

   0.0097823
 iter1  =

   2.
 n1  =

   4.
 II  =

   25.834565


 ea2  =

   0.0997471
 iter2  =

   2.
 n2  =

   4.
 I2  =

   1.941836
```