1. Solution for question 8.1

Let C_E, C_S, C_{ES} and C_P denote the concentration of E, S, ES and P, respectively. Then, their equations for the rate of changes are as follows.

$$\frac{dC_E}{dt} = k_2 C_{ES} + k_3 C_{ES} - k_1 C_E C_S,$$

$$\frac{dC_S}{dt} = k_2 C_{ES} - k_1 C_E C_S,$$

$$\frac{dC_{ES}}{dt} = k_1 C_E C_S - k_2 C_{ES} - k_3 C_E C_P,$$

$$\frac{dC_P}{dt} = k_3 C_{ES}.$$

2. Solution for question 8.2

For convenience, let $y_1=C_E$, $y_2=C_S$, $y_3=C_{ES}$, $y_4=C_P$, and $\mathbf{y} = (y_1, y_2, y_3, y_4)$. Then, rewrite the equation system to the following form:

$$\begin{cases} y_1'(t) = 750y_3 - 100y_1y_2 \\ y_2'(t) = 600y_3 - 100y_1y_2 \\ y_3'(t) = 100y_1y_2 - 600y_3 - 150y_1y_4 \\ y_4'(t) = 150y_3 \end{cases} where \begin{cases} y_1(0) = 1 \\ y_2(0) = 10 \\ y_3(0) = 0 \\ y_4(0) = 0 \end{cases}$$

For $i = 1, 2, 3$ and 4, y_i' can also be written as $f_i(t, \mathbf{y})$. Now, the whole system can be converted into a single ODE

$$\mathbf{y}'(t) = f(t, \mathbf{y}),$$

where $f(t, \mathbf{y}) = (f_1(t, \mathbf{y}), f_2(t, \mathbf{y}), f_3(t, \mathbf{y}), f_4(t, \mathbf{y}))$. Based on this equation, the formulas below can be used to solve it.

$$\begin{cases} y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = hf(x_i y_i) \\ k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\ k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\ k_4 = hf(x_i + h, y_i + k_3) \end{cases}$$

Since MATLAB is not on my laptop, I would use Java to write a programme that can compute the result when given the time interval and the number of partitions. The source code of the programme and sample output are as follows.

Sample output:

Please specify the end time and length of subinterval: 0.0021 0.000525
Current concentration of E, S, ES and P are:
0.666614, 9.650618, 0.349055, 0.015996
0.542216, 9.491641, 0.506562, 0.050575
0.502667, 9.408685, 0.586610, 0.093982

0.498379, 9.356163, 0.634512, 0.142216

Source code:

```
import java.math.BigDecimal;
import java.util.Scanner;

public class RKCalculator
{
    private static BigDecimal[] y = new BigDecimal[4]; // the vector of y
    private static double delta;      // length of subintervals
    private static double start_time;     // the start time for computation
    private static double end_time; // the end time for computation

    private static BigDecimal[] f(double t, BigDecimal[] v)
    {
        BigDecimal[] temp = new BigDecimal[4];

        temp[0] = v[2].multiply(new BigDecimal("750"))
                .subtract(v[0].multiply(v[1].multiply(new BigDecimal("100"))));
        temp[1] = v[2].multiply(new BigDecimal("600"))
                .subtract(v[0].multiply(v[1].multiply(new BigDecimal("100"))));
        temp[2] = v[0].multiply(v[1].multiply(new BigDecimal("100")))
                .subtract(v[2].multiply(new BigDecimal("600")))
                .subtract(v[0].multiply(v[3].multiply(new BigDecimal("150"))));
        temp[3] = v[2].multiply(new BigDecimal("150"));

        return temp;
    }

    private static void RK_Iteration()
    {
        System.out.println("Current concentration of E, S, ES and P are: ");
        for(double i=start_time; i<end_time; i+=delta)
        {
            // calculate k1
            BigDecimal[] k1 = multiply(Double.toString(delta), f(i, y));
            // calculate k2
            BigDecimal[] k2 = multiply(Double.toString(delta),
                    f(i+delta/2, addition(y, multiply("0.5", k1))));
            // calculate k3
            BigDecimal[] k3 = multiply(Double.toString(delta),
                    f(i+delta/2, addition(y, multiply("0.5", k2))));
            // calculate k4
            BigDecimal[] k4 = multiply(Double.toString(delta),
```

```
                    f(i+delta, addition(y, k3)));
            // calculate y
            y = addition(y, multiply(Double.toString(1.0/6.0), addition(k4,
                    addition(multiply("2", k3), addition(k1, multiply("2", k2))))));
            System.out.printf("%f, %f, %f, %f\n", y[0].doubleValue(),
            y[1].doubleValue(), y[2].doubleValue(), y[3].doubleValue());
        }
    }

    public static void main(String[] args)
    {
        // initialise y vector
        y[0] = new BigDecimal("1.0");
        y[1] = new BigDecimal("10.0");
        y[2] = new BigDecimal("0.0");
        y[3] = new BigDecimal("0.0");
        start_time = 0.0;  // initialise start time
        System.out.print("Please specify the end time and length of subinterval: ");
        Scanner scanner = new Scanner(System.in);
        end_time = scanner.nextDouble();
        delta = scanner.nextDouble();

        // invoke the iteration method to compute
        RK_Iteration();
    }

    private static BigDecimal[] multiply(String factor, BigDecimal[] v)
    {
        BigDecimal[] temp = new BigDecimal[4];
        for(int i=0; i<4; i++)
            temp[i] = v[i].multiply(new BigDecimal(factor));
        return temp;
    }

    private static BigDecimal[] addition(BigDecimal[] v1, BigDecimal[] v2)
    {
        BigDecimal[] temp = new BigDecimal[4];
        for(int i=0; i<4; i++)
            temp[i] = v1[i].add(v2[i]);
        return temp;
    }
}
```

3. Solution for question 8.3

To draw a graph, I used an online tool which is available at https://www.mathstools.com/section/main/runge_kutta_calculator#.YhpQUuhBy3A. The generated figure is shown below, where x_n denotes t, y_n is C_E, z_n is C_S, w_n is C_{ES} and u_n is C_P.