Q2

1.

8.1

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_3} E + P$$

$$e = [E] \quad s = [S] \quad c = [ES] \quad p = [P]$$

$$\dot{e} = -k_1 se + k_2 c + k_3 c$$

$$\dot{s} = -k_1 se + k_2 c$$

$$\dot{c} = k_1 se - k_2 c - k_3 c$$

$$\dot{p} = k_3 c$$

2.

Here is the code

```
import numpy as np

# Define the rate constants
k1 = 100 / 60
k2 = 600 / 60
k3 = 150 / 60

# Define the initial concentrations
E_0 = 1
S_0 = 10
ES_0 = 0
P_0 = 0

# Define the time range and step size
t_start = 0
t_end = 5
dt = 0.01

# Define the function that returns the derivatives of E, S, ES, and P
def derivatives(concentrations, t):
    E, S, ES, P = concentrations
```

```
    dE_dt = -k1 * ES + k2 * (E * S) / (E + S)
    dS_dt = -k1 * ES + k2 * (E * S) / (E + S)
    dES_dt = k1 * (E * S) / (E + S) - k2 * ES - k3 * ES
    dP_dt = k3 * ES
    return np.array([dE_dt, dS_dt, dES_dt, dP_dt])

# Define the fourth-order Runge-Kutta method
def rk4_step(f, y, t, dt):
    k1 = dt * f(y, t)
    k2 = dt * f(y + 0.5 * k1, t + 0.5 * dt)
    k3 = dt * f(y + 0.5 * k2, t + 0.5 * dt)
    k4 = dt * f(y + k3, t + dt)
    return y + (k1 + 2 * k2 + 2 * k3 + k4) / 6

# Define the array to store the results
t_array = np.arange(t_start, t_end + dt, dt)
concentrations_array = np.zeros((len(t_array), 4))

# Set the initial concentrations
concentrations_array[0, :] = [E_0, S_0, ES_0, P_0]

# Use the fourth-order Runge-Kutta method to solve the equations
for i in range(1, len(t_array)):
    concentrations_array[i, :] = rk4_step(derivatives,
concentrations_array[i-1, :], t_array[i-1], dt)

# Print the final concentrations
print(f"Final concentrations: E = {concentrations_array[-1, 0]:.3f}
µM, S = {concentrations_array[-1, 1]:.3f} µM, ES =
{concentrations_array[-1, 2]:.3f} µM, P = {concentrations_array[-1,
3]:.3f} µM")
```

Q3.

Here is the code

```
import matplotlib.pyplot as plt

t_start = 0
t_end = 5
dt = 0.01

# Define the function that returns the derivatives of E, S, ES, and P
def derivatives(concentrations, t):
    E, S, ES, P = concentrations
```

```
    dE_dt = -k1 * ES + k2 * (E * S) / (E + S)
    dS_dt = -k1 * ES + k2 * (E * S) / (E + S)
    dES_dt = k1 * (E * S) / (E + S) - k2 * ES - k3 * ES
    dP_dt = k3 * ES
    return np.array([dE_dt, dS_dt, dES_dt, dP_dt])

# Define the array to store the results
t_array = np.arange(t_start, t_end + dt, dt)
concentrations_array = np.zeros((len(t_array), 4))

# Set the initial concentrations
concentrations_array[0, :] = [E_0, S_0, ES_0, P_0]

# Use the fourth-order Runge-Kutta method to solve the equations
for i in range(1, len(t_array)):
    concentrations_array[i, :] = rk4_step(derivatives,
concentrations_array[i-1, :], t_array[i-1], dt)

# Calculate the velocity as the derivative of P with respect to time
P = concentrations_array[:, 3]
V = np.gradient(P, t_array)

plt.plot(concentrations_array[:, 1], V)
plt.xlabel('Concentration of substrate S (µM)')
plt.ylabel('Velocity V (µM/min)')
plt.show()

# Find the maximum value of V
Vm = np.max(V)
print(f"Maximum velocity Vm = {Vm:.3f} µM/min")
```

Here is the plot: