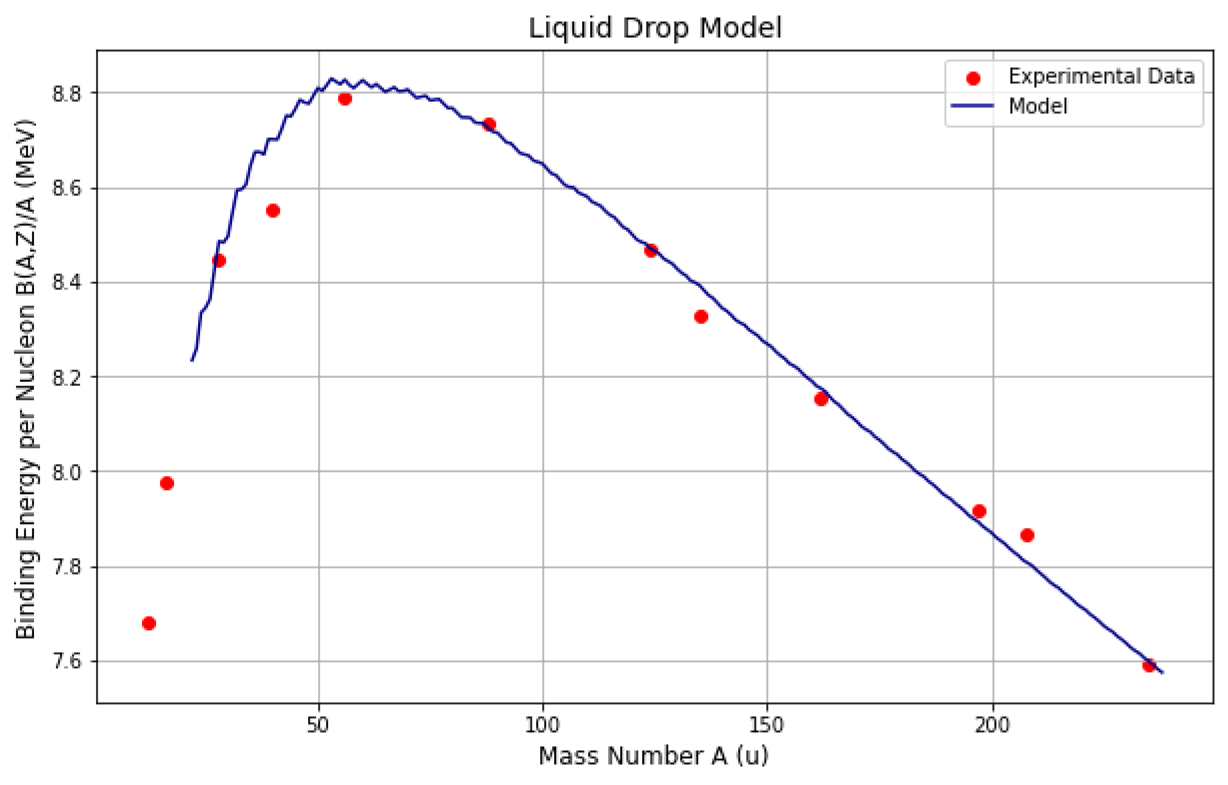
Problem 1

Using all binding energies determine the 5 constants of the liquid drop model

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


Z_exp = np.array([6,8,14,20,26,38,50,59,68,79,82,92])
N_exp = np.array([6,8,14,20,30,50,74,76,94,118,126,143])
B_AZ = np.array([92.16,127.6,236.5,342.0,492.3,768.5,1050,1124.4,1320.7,1559.4,1636.4,1783.9])
A_exp = Z_exp + N_exp

m_nc = 939.5527 # MeV
m_pc = 938.2592 # MeV

def Binding(A, a_V, a_s, a_c, a_A, a_p):

    # for a given A, the most stable Z needs to be determined
    num = (A / 2) + ((m_nc - m_pc) * (A / (8 * a_A))) + ((a_c / (8 * a_A)) * A**(2/3))
    den = den = 1 + (0.25 * (a_c / a_A) * A**(2/3))
    Z_stable = num / den
    Z_model = np.round(Z_stable) # round to integer

    # binding energy
    B_volume = a_V * A
    B_surface = a_s * A**(2/3)
    B_coulomb = a_c * ((Z_model * (Z_model - 1)) / (A**(1/3)))
    B_asymm = a_A * ((A - 2 * Z_model)**2) / A
    B_pair = (((-1)**Z_model + (-1)**(A - Z_model)) / 2) * (a_p / np.sqrt(A))

    B_total = B_volume - B_surface - B_coulomb - B_asymm + B_pair

    return B_total / A

initial_guess = [15.85, 18.34, 0.71, 23.21, 12.0]

params, cov = curve_fit(Binding, A_exp, B_AZ / A_exp, p0=initial_guess)

A = np.array(list(range(20, 241)))

plt.figure(figsize=(10,6))
plt.scatter(A_exp, B_AZ / A_exp, label="Experimental Data", color='red')

# total binding energy model
# between A = 20 to A = 100, the model oscillates rather than showing the average between the points
# due to the sign change of the pairing term
# plt.plot(A, Binding(A, *params), label="Model", color='mediumslateblue') to plot original model

plt.title("Liquid Drop Model", size=14)
plt.xlabel("Mass Number A (u)", size=12)
plt.ylabel("Binding Energy per Nucleon B(A,Z)/A (MeV)", size=12)
plt.grid()

# Total Binding energy model oscillates due to the changing sign of pairing term
# smoothed out the curve a little to see the model and experimental data better
kernel_size = 5
kernel = np.ones(kernel_size) / kernel_size

smoothed_model = np.convolve(Binding(A, *params), kernel, mode='same')

plt.plot(A[2:219], smoothed_model[2:219], color='darkblue', label='Model')
plt.legend()
plt.show()

# print each fitting parameter
print('Fitting Parameters:')
print('a_V -', round(params[0], 3))
print('a_s -', round(params[1], 3))
print('a_c -', round(params[2], 3))
print('a_A -', round(params[3], 3))
print('a_p -', round(params[4], 3))

# a_p is different from the initial guess while the other parameters are close to the guess
# probably because of smoothing the curve
```



```
Fitting Parameters:
a_V - 15.833
a_s - 17.832
a_c - 0.748
a_A - 22.136
a_p - 17.735
```

In [2]:
```
def Binding(A, Z):

    # binding energy
    # using parameters given

    B_volume = 15.85 * A
    B_surface = 18.34 * A**(2/3)
    B_coulomb = 0.71 * ((Z * (Z - 1)) / (A**(1/3)))
    B_asymm = 23.21 * ((A - 2 * Z)**2) / A
    B_pair = (((-1)**Z + (-1)**(A - Z)) / 2) * (12 / np.sqrt(A))

    B_total = B_volume - B_surface - B_coulomb - B_asymm + B_pair

    return B_total

print('B(A,Z) Ba -', Binding(106, 46), 'MeV')
print('B(A,Z) Kr -', Binding(238, 92), 'MeV')
print('B(A,Z) U -', Binding(235, 92), 'MeV')
```

```
B(A,Z) Ba - 917.0280891293778 MeV
B(A,Z) Kr - 1825.1947078991814 MeV
B(A,Z) U - 1806.2148938523962 MeV
```

In [ ]: