Question 1

```
import numpy as np


# function for calculating factorial
def factorial(n):
    val = 1;
    for i in range(2, n + 1):
        val = val * i;

    return val;

# function for calculating u values
def u_values(u, n):
    val = u;
    for i in range(1, n):
        val = val * (u - i);
    return val;

# forward newton raphson function
def forward_newton_method(X, Y, input_x):
  n = X.shape[0]

  # initialising the value of interpolated y to 0
  calc_y = Y[0][0]

  # finding the forward differences and storing them in Y matrix
  for i in range(1, n):
      for j in range(n - i):
          Y[j][i] = Y[j + 1][i - 1] - Y[j][i - 1];

  # initial value of u
  u = (input_x - X[0]) / (X[1] - X[0])
  for i in range(1,n):
      calc_y = calc_y + (u_values(u, i) * Y[0][i]) / factorial(i);

  return calc_y

# taking the actual function as y = x^3 and giving 2 inputs initially
X = np.array([10, 30])

# array Y will store the y values and forward difference values
Y = np.zeros((2,2))
```

```
# initialising the first entries of every row in Y to the y values they take
Y[0][0], Y[1][0] = 1000, 27000

input_x = 25
actual_y = 15625
calc_y = forward_newton_method(X, Y, input_x)

# printing output
print("f(x) at x = {} is: {}".format(input_x, calc_y))
print("error percentage is:", 100*abs(actual_y - calc_y)/actual_y)
# output below
f(x) at x = 25 is: 20500.0
error percentage is: 31.2


# now checking if error percentage decreases by adding more terms

X = np.array([10, 20, 30])
Y = np.zeros((3,3))
Y[0][0], Y[1][0], Y[2][0] = 1000, 8000, 27000

input_x = 25
actual_y = 15625
calc_y = forward_newton_method(X, Y, input_x)

# printing output
print("f(x) at x = {} is: {}".format(input_x, calc_y))
print("error percentage is:", 100*abs(actual_y - calc_y)/actual_y)
# output below
f(x) at x = 25 is: 16000.0
error percentage is: 2.4


# increasing  2 more terms to check if error percentage goes down even more

n = 5
X = np.array([10, 20, 30, 40, 50])
Y = np.zeros((5,5))
Y[0][0], Y[1][0], Y[2][0], Y[3][0], Y[4][0] = 1000, 8000, 27000, 64000, 125000

input_x = 25
actual_y = 15625
calc_y = forward_newton_method(X, Y, input_x)
```

```
# printing output
print("f(x) at x = {} is: {}".format(input_x, calc_y))
print("error percentage is:", 100*abs(actual_y - calc_y)/actual_y)
# output below
f(x) at x = 25 is: 15625.0
error percentage is: 0.0
```

Question 2

```
# function to calculate polynomial p2(x)
def lagrange_interpolation(X, Y, input_x):
  n = X.shape[0]
  p2 = 0

  for i in range(n):
    L = 1
    for j in range(n):
      if i != j:
        L = L * (input_x - X[j])/(X[i] - X[j])
    p2 = p2 + L*Y[i]

  return p2

# known values of (x,y)
X = np.array([0.25, 0.5, 1])
Y = np.array([0.27633, 0.52050, 0.84270])

# value of x at which we have to interpolate
input_x = 0.75
calc_y = lagrange_interpolation(X, Y, input_x)

# Displaying output
print("Value of interpolated f(x) at x = 0.75 is:", calc_y)
# Output below
Value of interpolated f(x) at x = 0.75 is: 0.70929
```