## Question 1

```
import numpy as np

def matrix_mul(B, C):
  return np.matmul(B, C)

B = np.array([[2,5,-2],[-1,0,0],[2,3,4]])
C = np.array([[3,5],[1,0],[2,0]])

A = matrix_mul(B, C)
print(A)
# output below
[[ 7 10]
 [-3 -5]
 [17 10]]


# another function to implement matrix multiplication
def matrix_mul_alt(B, C):
  A = np.zeros((B.shape[0], C.shape[1]))
  for i in range(len(B)):
    for j in range(C.shape[1]):
      for k in range(C.shape[0]):
        A[i][j] += B[i][k] * C[k][j]

  return A

B = np.array([[2,5,-2],[-1,0,0],[2,3,4]])
C = np.array([[3,5],[1,0],[2,0]])
A = matrix_mul_alt(B, C)
print(A)
# output below
[[ 7. 10.]
 [-3. -5.]
 [17. 10.]]
```

## Question 2

Back substitution is a mathematical technique that can be used to solve an upper triangular linear system of equations, i.e., if the system is Ax = b, and A is an upper triangular matrix, then using back substitution we can get all the x's. In the example given, A is a upper triangular matrix. So a33 * x3 = b3. From here, we get x3. Now, we put x3 in the equation formed using the second row:

a22 * x2 + a23 * x3 = b2.

x2 = (b2 - a23*x3)/a22

Now since we know x3, we can get x2 from the above equation as it is the only unknown.
Now we have x1 and x2. We can substitute this into the equation formed using the first row of A:

a11 * x1 + a12 * x2 + a13 * x3 = b1.

x1 = (b1 - a12*x2 - a13*x3)/a11

Thus we got all the unknowns by starting from the lowest row and substituting the unknowns found at each step as we go up the rows. This is the technique of back substitution.

```
# back substitution algorithm
def back_subs(A, b):
  n = A.shape[1]
  x = np.zeros(n)

  # the value of the first unknown (x3 in our case)
  x[n-1] = b[n-1]/A[n-1][n-1]

  for i in range(n-2,-1,-1):
    x[i] = b[i]
    for j in range(i+1, n):
      x[i] -= A[i][j] * x[j]

    x[i] /= A[i][i]

  return x

# inputs
A = np.array([[2,-3,1],[0,-3,-1],[0,0,5]])
b = np.array([-1,-9,15])
x = back_subs(A, b)

# printing the values of unknowns
for i in range(len(x)):
  print("x{} = {}".format(i+1, x[i]))

# output below
x1 = 1.0
```

```
x2 = 2.0
x3 = 3.0
```