```
def matrix_multiply_sequential(A, B):
    if len(A[0]) != len(B):
        raise ValueError("Matrix dimensions are incompatible for multiplication")

    m = len(A)
    n = len(A[0])
    p = len(B[0])

    result = [[0 for _ in range(p)] for _ in range(m)]

    for i in range(m):
        for j in range(p):
            temp_sum = 0
            for k in range(n):
                temp_sum += A[i][k] * B[k][j]
            result[i][j] = temp_sum

    return result



import threading

def matrix_multiply_one_thread_per_cell(A, B):
    if len(A[0]) != len(B):
        raise ValueError("Matrix dimensions are incompatible for multiplication")

    m = len(A)
    n = len(A[0])
    p = len(B[0])

    result = [[0 for _ in range(p)] for _ in range(m)]
    threads = []

    def multiply_cell(i, j):
        temp_sum = 0
        for k in range(n):
            temp_sum += A[i][k] * B[k][j]
        result[i][j] = temp_sum

    for i in range(m):
        for j in range(p):
            thread = threading.Thread(target=multiply_cell, args=(i, j))
            threads.append(thread)
            thread.start()

    for thread in threads:
        thread.join()

    return result



import threading

def matrix_multiply_one_thread_per_row(A, B):
    if len(A[0]) != len(B):
        raise ValueError("Matrix dimensions are incompatible for multiplication")

    m = len(A)
    n = len(A[0])
    p = len(B[0])

    result = [[0 for _ in range(p)] for _ in range(m)]
    threads = []

    def multiply_row(row):
        for j in range(p):
            temp_sum = 0
            for k in range(n):
                temp_sum += A[row][k] * B[k][j]
            result[row][j] = temp_sum

    for i in range(m):
        thread = threading.Thread(target=multiply_row, args=(i,))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()
```

```
    return result


import random

# Function to generate a random matrix of size m x n with values between 1 and 10
def generate_random_matrix(m, n):
    return [[random.randint(1, 10) for _ in range(n)] for _ in range(m)]

# Define matrix dimensions
m = 3  # Number of rows
n = 4  # Number of columns
p = 2  # Number of columns

# Generate random matrices A and B
A = generate_random_matrix(m, n)
B = generate_random_matrix(n, p)

# Call sequential matrix multiplication
result_sequential = matrix_multiply_sequential(A, B)

# Call multithreaded matrix multiplication with one thread per row
result_multithreaded_row = matrix_multiply_one_thread_per_row(A, B)

# Call multithreaded matrix multiplication with one thread per cell
result_multithreaded_cell = matrix_multiply_one_thread_per_cell(A, B)

# Print the results
print("Sequential Matrix Multiplication Result:")
for row in result_sequential:
    print(row)

print("\nMultithreaded Matrix Multiplication (One Thread Per Row) Result:")
for row in result_multithreaded_row:
    print(row)

print("\nMultithreaded Matrix Multiplication (One Thread Per Cell) Result:")
for row in result_multithreaded_cell:
    print(row)
```

```
Sequential Matrix Multiplication Result:
[94, 227]
[94, 229]
[94, 269]

Multithreaded Matrix Multiplication (One Thread Per Row) Result:
[94, 227]
[94, 229]
[94, 269]

Multithreaded Matrix Multiplication (One Thread Per Cell) Result:
[94, 227]
[94, 229]
[94, 269]
```