

```

static void matrixMult (int n, float[][] a,
    float[][] b, float[][] prod) {
// Set prod to the product of the n×n matrices a and b.
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            float s = 0.0;
            for (int k = 0; k < n; k++) {
                s += a[i][k] * b[k][j];
            }
            prod[i][j] = s;
        }
    }
}

```

Analyze these methods in terms of the number of **float** additions and multiplications performed. What is the time complexity of each method?

- 2.5 Analyze the time complexity of the recursive algorithm to render a given integer i to base r (Algorithm 2.16).
- 2.6 Devise a non-recursive algorithm to print a given integer i to base r . What is your algorithm's time complexity?
- 2.7 Devise a recursive version of Euclid's algorithm to calculate the GCD of two numbers. (A non-recursive version of this algorithm was given in Algorithm 1.3.)
- 2.8 The factorial of a positive integer n can be calculated using the recursive algorithm given in Algorithm 2.21. What is the time complexity of this algorithm?
Devise a non-recursive version of this algorithm.
Implement both algorithms as Java methods.
- * 2.9 The Fibonacci number of a non-negative integer n can be calculated using the recursive algorithm given in Algorithm 2.22. What is the time complexity of this algorithm?
Devise a non-recursive version of this algorithm.

To calculate the factorial of n :

1. If $n = 0$:
 - 1.1. Terminate with answer 1.
2. If $n \neq 0$:
 - 2.1. Let f be the factorial of $n-1$.
 - 2.2. Terminate with answer $(n \times f)$.

Algorithm 2.21 A recursive factorial algorithm.