Exercise 1:

Below is the pseudocode for an algorithm that calculates the n^{th} fibonacci number, assuming that arrays are counted from 1.

```
Fibonacci(n):
    if (n < 3):
        return 1;
    else:
        int arr[n];
        arr[1] = 1;
        arr[2] = 1;
        for (int i = 3; i <= n; i++):
            arr[i] = arr[i-1] + arr[i-2];
        return arr[n];
```

a. What is the input size of the above algorithm?
b. What is the best-case scenario?
c. What is the worst-case scenario?
d. Identify the elementary operation(s) then calculate the running time for the worst-case instance.

Exercise 2:

Below is the pseudocode for an algorithm that checks if the sum of all the elements in an array is divisible by every single element of that array, assuming that arrays are counted from 1.

```
Divisible(A[1..n]):
    int sum = 0;
    for (int i = 1; i <= n; i++):
        sum += A[i];
    for (int i = 1; i <= n; i++):
        if (sum % A[i] != 0):
            return False;
    return True;
```

a. What is the input size of the above algorithm?
b. What is the best-case scenario?
c. What is the worst-case scenario?
d. Identify the elementary operation(s) then calculate the running time for the worst-case instance.

Exercise 3:

Below is the pseudocode for the Floyd-Warshall algorithm (that you will learn eventually), assuming that arrays are counted from 1. For now you don't have to fully understand what this algorithm does. You only need to pay attention to the running time.

```
FLOYD-WARSHALL(W)
1. n ← rows[W]
2. D^(0) ← W
3. for k ← 1 to n
4.       do for i ← 1 to n
5.             do for j ← 1 to n
6.                   d_ij^(k) ← min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
7. return D^(n)
```

a. What is the input size of the above algorithm?
b. Identify the elementary operation(s) then calculate the running time for the worst-case instance.

Exercise 4:

Given this pseudocode:

```
DaAlgorithm(A[1..n], B[1..m]):
    for (int i = 1; i <= n; i++):
        for (int j = 1; j <= n-1; j++):
            print(i*j);
```

```
for (int i = 1; i <= m; i++):
        for (int j = 1; j <= 10; j++):
                for (k = 1; k <= n; k++):
                        print(i*j*k);
return;
```

a. What is the input size of the above algorithm?
b. Identify the elementary operation(s) then calculate the running time for the worst-case instance.