Suppose module A requires M units of time to be executed, where M is a constant. Find the complexity C(n) of the following algorithm, where n is the size of the input data. (5)

```
Algorithm: 1.  Repeat for I = 1 to N:
           2.      Repeat for J = 1 to N:
           3.          Repeat for K = 1 to N:
           4.              Module A.
                       [End of Step 3 loop.]
                   [End of Step 2 loop.]
               [End of Step 1 loop.]
           5.  Exit.
```

Solution:

The innermost loop executes the module M

$$C(n) = \sum_{K=1}^{n} M \quad \text{times.}$$

The outer loop executes n times that results in the execution of the module M

$$C(n) = \sum_{J=1}^{n} \sum_{K=1}^{n} M \quad \text{times.}$$

The outermost loop executes n times, that results in the execution of the module M

$$C(n) = \sum_{I=1}^{n} \sum_{J=1}^{n} \sum_{K=1}^{n} M$$

The number of times M occurs in the sum is equal to the number of triplets (I, J, K), where I, J and K are integers from 1 to n inclusive. There are n^3 such triplets. Hence

$$C(n) = M \times n^3 = O(n^3)$$

Assume that the input size is n, and for the running time we will count the number of time that any element of P is accessed. Clearly we go through the outer loop n times, and for each time through this loop, we go through the inner loop n times as well. The condition in the if-statement makes four accesses to P. The output statement makes two accesses for each point that is output. In the worst case every point is maximal (can you see how to generate such an example?) so these two access are made for each time through the outer loop.

```
MAXIMA(int n, Point P[1...n])
for i ← 1 to n                                        n times
do maximal ← true
    for j ← 1 to n                                    n times
    do
        if (i ≠ j)&(P[i].x ≤ P[j].x)&(P[i].y ≤ P[j].y)  4 accesses
          then maximal ← false break
    if maximal
      then output P[i].x, P[i].y                       2 accesses
```

Thus we might express the worst-case running time as a pair of nested summations, one for the i-loop and the other for the j-loop:

$$T(n) = \sum_{i=1}^{n} (2 + \sum_{j=1}^{n} 4$$

$$= \sum_{i=1}^{n} (4n + 2)$$

$$\mathbf{= (4n + 2)n = 4n^2 + 2n}$$