Problem 9:

a) The outer for loop runs for at most n iterations which is in O(n) while the inner for loop run for at most n iterations, which is also O(n). Then the summation of array entries in the inner for loop takes at most n steps which is also O(n). The other operations like assigning values and storing them in array takes constant time. Hence over all the algorithm takes all most O(n)·O(n)·O(n) steps which equals to $O(n^3)$ so f(n) = $O(n^3)$

b) Let's consider the first n/3 iterations where i = $\frac{n}{3}$

Then the second loop runs for at least (n - $\frac{n}{3}$) = $\frac{2n}{3}$

The summation takes place from A[i] to A[j] so it takes at least $\frac{2n}{3} - \frac{n}{3} = \frac{n}{3}$

As a result, there are at least $(\frac{n}{3})^3 = \frac{1}{27}n^3$ steps involved in summation.

$\frac{1}{27}n^3 \leq n^3$

Hence algorithm is also in Ω (n^3)

c)

```
For i=1 to n-1 do
        temp = A[i]
        for j=i+1 to n do
                temp = temp + A[j]
                B[i][j] = temp
        EndFor
EndFor
Return B
```

The outer loop runs for exactly n iterations while the inner for loop takes at most n iterations so total time for the algorithm considering the other operations take constant time is n·n = n^2
Hence f(n) = $O(n^2)$