**2.1**

Om vi ordnar funktionerna med avseende på tillväxttakt med den med lägst tillväxttakt först fås ordningen:

* *f4(n)* = *n* +100
* *f3(n)* = *n* log *n*
* *f1(n)* = *n*1.5
* *f5(n)* = 2*n*
* *f2(n)* = 10*n*

1. *n*(*n* + 1) / 2 = *O*(*n*3)
2. *n*(*n* + 1) / 2 = *O*(*n*2)
3. *n*(*n* + 1) / 2 = Θ(*n*3)
4. *n*(*n* + 1) / 2 = Ω(*n*)

Påstående ett är sant då V.L växer långsammare än n^3, dvs. V.L är O(n^3).

Påstående 2 är sant då n^2 är en övre begränsning för stora n.

Påstående 3 är falskt då n^3 ej är en undre begränsning även fast n^3 är en övre begränsning.

Påstående 4 är sant då n är en undre begränsning.

**2.2**

Algoritmen är O(n^2) då vi i värstafallet får två nästlade for-loopar från 0 till (n-1).

Vi har även att n^2 är en undre begränsning till algoritmen vilket kan inses genom att kontrollera för små n:

För n = 1 utförs 3 operationer, 3> n^2=1. För n = 2 utförs 7 operationer, 7 > n^2=4 osv. Alltså är n^2 en undre begränsning och vi kan då säga att algoritmen är Omega(n^2).

En förbättrad algoritm:

B[0] = A[0]

for i = 1 to n-1

B[i] = B[i-1] + A[i]

Denna algoritm är Θ(n) eftersom den är både uppåt och nedåt begränsad av n.

**2.4**

Ett exempel på en ovanlig funktion som varken är Omega(n) eller O(n) är tex. n^2\*abs(sin(pi\*n/2)) som kommer vara antingen 0 eller n^2 beroende på n. är n ingen övre begränsning men n är inte heller någon undre begränsning.