1.13:

(a) Let c = 20, 17 < 1 * 20, $\forall$ n ≥ 1. Therefore, 17 is O(1).

(b) Let c = 4, n(n-1)/2 ≤ 4 * n^2, n^2 - n ≤ $8n^2$, $\forall$ n > 0. Therefore, n(n-1)/2 is O(n2).

(c) Let c = 1, max(n^3, $10n^2$) ≤ 1 * n^3, $\forall$ n > 10. Therefore, max(n3, 10n2) is O(n3).

(d) Σ(1..n) i^k = (n^{k+1}-1) / (k+1).

Let c = 1 / (k+1) , (n^{k+1}-1) / (k+1) ≤ (n^{k+1}) / (k+1), $\forall$ n ≥ 1. Therefore, Σ(1..n) i^k is O(n^{k+1}).

Let c = 1 / (k+3) , (n^{k+1}-1) / (k+1) ≥ (n^{k+1}) / (k+3), $\forall$ n ≥ 1. Therefore, Σ(1..n) i^k is Ω(n^{k+1}).

(e) k^{th} polynomial P(n) = $a_k n^k + a_{k-1} n^{k-1}$+ ... .. + $a_{0.}$

Let c = a_n + a_{n-1} + ... a_o, P(n) ≤ (a_n + a_{n-1} + ... a_o) * n^k, $\forall$ n ≥ 1. Therefore P(n) is O(n^k).

Let c = 1, P(n) ≥ 1 * n^k, $\forall$ n > 1. Therefore P(n) is Ω(n^k).

1.16: $(1/3)^n$ < 17 < loglogn < logn < $\log^2 n$ < √n < √n$\log^2$n < n/logn < n <$(3/2)^n$

1.18:

(a) T(n) = O(1) + T(n/2) + T(n/2) = O(1) + 2T(n/2)

(b) T(n) = i * 2 * logn, $\forall$ i > 0. Let c = 1, 2logn ≤ n, $\forall$ n > 0. Therefore T(n) is Ω(n).

2.9: If the list is empty, the code will still execute and occur errors. To fix this problem, add an if statement at the beginning of the program to check if the list is empty.

2.11:

```
p := FIRST(L);                          1 time
while p <> END(L) do begin              n times
        q := p;
        while q <> END(L) do begin      n times
                q := NEXT(q, L);        n - q times
                r := FIRST(L);          1 time
                while r <> q do
                        r := NEXT(r, L)n - r times
                end;
        p := NEXT(p, L)                 n - p times
end;
```

FIRST: 1 * n * n + 1 = n^2 + 1 times

NEXT: (n - q) * (n - r) * (n - p) * n times

END: n * n = n^2 times