Problem 1.6

(2)

$$\sum_{i=1}^{n}\sum_{j=i}^{n} 1 = \sum_{i=1}^{n}(n+1-i) = \frac{n(n+1)}{2}$$

$$f(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2}, \quad g(n) = n^2$$

(5)

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=1}^{1000} 1 &= \sum_{i=1}^{n}\sum_{j=i}^{n} 1000 \\
&= 1000\left(\sum_{i=1}^{n}(n+1-i)\right) \\
&= 1000\left((n)(n+1) - \frac{n(n+1)}{2}\right) \\
&= 1000\left(\frac{n(n+1)}{2}\right) \\
&= 500n(n+1)
\end{aligned}$$

$$f(n) = 500n(n+1), \quad g(n) = n^2$$

(6)

$$\begin{aligned}
\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{j} 1 &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} j \\
&= \sum_{i=1}^{n-1}\frac{(n-i)(n+i+1)}{2} \\
&= \frac{1}{2}\sum_{i=1}^{n-1} n^2+n-i^2-i \\
&= \frac{1}{2}\left[(n-1)(n^2+n) - \frac{(n-1)(n)(2n-1)}{6} - \frac{(n-1)(n)}{2}\right] \\
&= \frac{1}{2}\left[\frac{(n-1)(4n^2+4n)}{6}\right] \\
&= \frac{1}{3}n(n^2-1)
\end{aligned}$$

$$f(n) = \frac{1}{3}n(n^2-1), \quad g(n) = n^3$$

Problem 1.8 (2)

```
poly = 0;
for (i = n; i >= 0; i--)
  poly = x * poly + a_i
```

The algorithm first initializes `poly` to 0 and uses a loop to evaluate the polynomial from n to 0, meaning it processes the terms in reverse order, from a_n to a_0. However, the algorithm works correctly because it multiplies by x each time i decreases. Since the loop runs n times, the initial coefficient a_n, after being assigned to `poly` at the beginning, is multiplied by x exactly n times, resulting in the term $a_n x^n$ when the loop ends. Additionally, the loop adds each coefficient a_i as i decreases, and each term becomes the multiplicand, getting multiplied by x $n-i$ times. This approach produces the desired outcome: $f(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0 x^0$.

Problem 1.9 (3)

Since $C(n) = t(n) + 5 * s(n)$, we have:

$$C_A(n) = \begin{cases} n^2 + 5n, & \text{if } 1 \leq n < 10; \\ 6n, & \text{if } 10 \leq n < 20; \\ 8.5n, & \text{if } 20 \leq n < 50; \\ n^3 + 7.5n, & \text{if } 50 \leq n \leq 100. \end{cases}$$

$$C_B(n) = \begin{cases} 26n, & \text{if } 1 \leq n < 30; \\ n^2 + 25n, & \text{if } 30 \leq n < 50; \\ n^2 + 2.5n, & \text{if } 50 \leq n < 70; \\ n^3 + 2.5n, & \text{if } 70 \leq n \leq 100. \end{cases}$$

Given the cost for different cases varies, which algorithm is better can be viewed as following:

	$C_A(n)$	$C_B(n)$	Better Algorithm
$1 \leq n < 10$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$C_B(n)$
$10 \leq n < 20$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$C_A(n)$
$20 \leq n < 30$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$C_A(n)$
$30 \leq n < 50$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$C_A(n)$
$50 \leq n < 70$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$C_B(n)$
$70 \leq n \leq 100$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$C_B(n)$

Problem 2.32

Operation	top(S)
add(4,S)	4
add(1,S)	1
add(3,S)	3
delete(S)	1
add(8,S)	8
delete(S)	1

Problem 2.33

Operation	front(Q)	rear(Q)
add(4,Q)	4	4
add(1,Q)	4	1
add(3,Q)	4	3
delete(Q)	1	3
add(8,Q)	1	8
delete(Q)	3	8

- END -