1. **Asymptotic notations:**

(a) Solve the recurrence: $T(n) = \frac{1}{n} + T(n-1)$

$$T(n) = \frac{1}{n} + T(n-1)$$
$$T(n-1) = \frac{1}{n-1} + T(n-2)$$
$$\cdots$$
$$T(1) = \frac{1}{2} + T(0)$$

Thus

$$T(n) = \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{1}{2} + 1 \sim \ln n + \gamma$$

(b) Prove or disprove:

(1) $2^{n+1} = O(2^n)$

Let $g(n) = 2^n$
$\exists c = 4, n_0 = 1, \forall n \geq n_0, f(n) = 2^{n+1} \leq c \cdot 2^n = 4 \cdot 2^n = 2^{n+2} = c \cdot g(n)$
The statement is proved.

(2) $2^{2n} = O(2^n)$

$\lim \frac{2^{2n}}{2^n} = \lim 2^n = \infty$
$\forall n_0 = c$ such that $\forall n \geq n_0$ $f(n) = 2^{2n} = 2^{2c} > c \cdot 2^c = c \cdot g(n)$
$\therefore$ $\nexists c, n_0$ satisfying $\forall n \geq n_0$ $f(n) \leq c \cdot g(n)$
$\therefore$ statement disproved

(c) Let P be a problem. The worst-case time complexity of P is $O(n^2)$.

The worst-case complexity of P is also $\Omega(n \log n)$. Let A be an algorithm that solves P. Which statements of the following statements are consistent with this formation about the complexity of P?

(1) A has worst-case time complexity $O(n^2)$

Consistent

(2) A has worst-case time complexity $O\left(n^{\frac{3}{2}}\right)$

Inconsistent

(3) A has worst-case time complexity $O(n)$

Inconsistent

(4) A has worst-case time complexity $\theta(n^2)$

Inconsistent

(5) A has worst-case time complexity $\theta(n^3)$

Inconsistent

2. Let $f(n) = O(s(n))$ and $g(n) = O(t(n))$. Prove or disprove the following.

$f(n) = O(s(n)) \Leftrightarrow s(n) = \Omega(f(n))$

$g(n) = O(t(n)) \Leftrightarrow t(n) = \Omega(g(n))$

(a) $f(n) + g(n) = O(s(n) + t(n))$

$f(n) = O(s(n)) \Rightarrow \exists c_1, n_1$ such that $\forall n \geq n_1, f(n) \leq c_1 \cdot s(n)$

$g(n) = O(t(n)) \Rightarrow \exists c_2, n_2$ such that $\forall n \geq n_2, g(n) \leq c_2 \cdot t(n)$

$\therefore$ Let $c_3 = \max(c_1, c_2), n_3 = \max(n_1, n_2)$
such that $\forall n \geq n_3, f(n) + g(n) \leq c_3 \cdot s(n) + c_3 t(n)$
$\therefore$ statement proved

(b) $s(n) * t(n) = \Omega(f(n) * g(n))$

$s(n) = \Omega(f(n)) \Rightarrow \exists c_1, n_1$ such that $\forall n \geq n_1, s(n) \geq c_1 \cdot f(n)$

$t(n) = \Omega(g(n)) \Rightarrow \exists c_2, n_2$ such that $\forall n \geq n_2, t(n) \geq c_2 \cdot g(n)$

$\therefore$ Let $c_3 = \min(c_1, c_2), n_3 = \max(n_1, n_2)$
such that $\forall n \geq n_3, s(n) * t(n) \geq c_3 \cdot f(n) \cdot c_3 \cdot g(n)$
$= c_3^2 \cdot f(n) \cdot g(n)$
$\therefore$ statement proved

(c) $f(n) - g(n) = O(s(n) - t(n))$

Let $f(n) = n^2, g(n) = n, s(n) = n^2, c(n) = n^2$

$f(n) - g(n) = O(n^2) \neq O(s(n) - t(n)) = 1$

Counterexample found.
$\therefore$ statement disproved

(d) $\frac{s(n)}{t(n)} = \Omega(\frac{f(n)}{g(n)})$

Let $s(n) = n^2, t(n) = n, f(n) = n, g(n) = n$

$\frac{s(n)}{t(n)} = \frac{n^2}{n} = n = \Omega(n) \neq 1 = \Omega\left(\frac{n}{n}\right) = \Omega(\frac{f(n)}{g(n)})$

Counter example found
$\therefore$ statement disproved

3. Let A and B be sets such that each has n positive integers in a non-decreasing order. We want to compute the set C such that $a \in C$ if and only if a appears either (i) in both A and B, or (ii) more than once in A or B. For example, if $A = \{1, 3, 3, 5, 7\}$ and $B = \{2, 4, 5, 6, 8\}$, then $C = \{3, 5\}$. Give an $O(n)$ time algorithm for the problem.

Description:

Empty element is different from any other element.

1) Initialize three pointers pointing to the very beginning of A,B and C(empty)
2) Compare the pointed element in A and B. If they are equal, then add it into C while place right after the pointed element in C when the pointed element in C is different from the element trying to add. Move all pointer forward by 1.

   If the pointed element in A is larger than the pointed element in B. Then check the next element in B. If the pointed element in B is the same as the next element in B, then compare it with the pointed element in C. Add the element right after the pointed element in C and move pointer in B and C forward by 1 only when element trying to add is different from the element pointed in C. Otherwise, just move the pointer forward by 1 in B.

   If the pointed element in A is smaller than the pointed element in B. Then check the next element in A. If the pointed element in A is the same as the next element in A, then compare it with the pointed element in C. Add the element right after the pointed element in C and move pointer in A and C forward by 1 only when the element trying to add is different from the element pointed in C. Otherwise, just move the pointer forward by 1 in A.

   Repeat this step until the pointer in A and B reach the end of the set.
3) Output C.

Matlab Code:

```
A=[1,2,2,3,5,6,8,9,9,9];
B=[0,2,2,3,5,6,7,8,9,10];
tempC(1)=NaN;
%Initialize the first element in C to be NaN (not a number)
szA=max(size(A));
szB=max(size(B));
% A and B are two set/array in non-decreasing order
Apointer=1;
Bpointer=1;
Cpointer=1;
```

```
while Apointer<=szA && Bpointer<=szB
    if A(Apointer)==B(Bpointer)
        temp_out=A(Apointer);
        if temp_out~=tempC(Cpointer)
            Cpointer=Cpointer+1;
            tempC(Cpointer)=temp_out;
            Apointer=Apointer+1;
            Bpointer=Bpointer+1;
        else
            Apointer=Apointer+1;
            Bpointer=Bpointer+1;
        end
    end
    if A(Apointer)<B(Bpointer)
        if Apointer==szA
            break
        end
        if A(Apointer)==A(Apointer+1)
            temp_out=A(Apointer);
            if temp_out~=tempC(Cpointer)
                Cpointer=Cpointer+1;
                tempC(Cpointer)=temp_out;
                Apointer=Apointer+1;
            else
                Apointer=Apointer+1;
            end
        else
            Apointer=Apointer+1;
        end
    end
    if A(Apointer)>B(Bpointer)
        if Bpointer==szB
            break
        end
        if B(Bpointer)==B(Bpointer+1)
            temp_out=B(Bpointer);
            if temp_out~=tempC(Cpointer)
                Cpointer=Cpointer+1;
                tempC(Cpointer)=temp_out;
                Bpointer=Bpointer+1;
            else
                Bpointer=Bpointer+1;
            end
        else
```

```
            Bpointer=Bpointer+1;
        end
    end
end
szC=max(size(tempC));
C=tempC(2:szC)
% output C in the console.
```