Assignment 3 plus practice questions

This is an individual/group homework assignment. Groups of **up to three** students can submit joint solutions. For group submissions, please follow the following rules:

Notes regarding groups (If you are submitting the work individually, please skip this part):
For group submissions, please follow the following rules:

1) **Form a Group:** First, self-enroll in a group with your partner(s) by navigating to People → Groups → **Assignment3 Group.**
2) **Re-enrollment Required:** Even if you were part of a group before, you must sign up for the Assignment3 Group again, as it is a separate tab under the Groups section.
3) **Submit as a Group**: When submitting the assignment, select the "Group Submission" option. This allows one team member to submit the work on behalf of the entire group, ensuring everyone in the group sees the grade and feedback. Important: **ALL GROUP PARTNER NAMES** should appear on ALL submitted documents.

You may use the Canvas Discussion Board to look for groupmates and create your group.

As it appears in the course syllabus, for the homework assignments, students are encouraged to discuss the problems with others, but you are expected to turn in the results of your own effort (not the results of a friend's efforts)". Even when not explicitly asked, you are supposed to justify your answers concisely.

Question 1) Write a complete program in either C++ or Java that counts the number of duplicates in a list of integer numbers. Your program receives a list of n numbers and outputs the total number of duplicates. The list is not sorted. Your algorithm should be of the order of $\theta(n^2)$. Analyze your algorithm in detail. Justify why it is $\boldsymbol{\theta(n^2)}$**.**

Example: Consider List = {1, 2, 1, 3, 3, 4, 3}; It has this be one duplicate of 1 and two duplicates of 3, therefore total duplicates will be three. The total will be 1+2=3 duplicates.

Question 2) Consider the previous question, but this time assume you are given a **sorted list** of numbers. Now consider the sorted list and write a code to count the number of duplicates again. Present your algorithm by pasting your new code in the solution file. Analyze the time complexity of your algorithm in the worst case. Could you improve the worst-case time complexity of your algorithm compared to the previous question? Why? Explain.

Note: When addressing coding questions, please adhere to the following guidelines: (i) Paste the entire code within the solution file and provide an analysis as requested. (ii) Additionally, submit your code files along with your solution. Please note that coding is only permitted in C++/Java; the use of other programming languages, such as Python, is not accepted and will result in a zero score for the question. Failure to submit the required code files will result in a 50% deduction from the grade obtained for the related question.

Question 3)
Consider the following functions as time complexities of some algorithms.

(a) First, write the worst-case runtime of each algorithm in Big-O notation.
(b) Then arrange functions from low to high (Consider their Big-O notations as you arrange them; as n grows, the function that grows slower should come sooner than the one that grows faster).
(c) Arbitrary, select **TWO** arrangements in part (b), and for each one of them justify why you have chosen such an ordering for the corresponding function. Provide formal proof. For example, if f1<=f2<= f3<= f4<=f5 you may choose to formally justify why (1) f1<=f2 and why (2) f4<=f5 . Show all your work and prove formally as we studied.

$f1(n)=15n^3$,
$f2(n)=10^{10}\ n\log n +2n$,
$f3(n)=3^n + 2n+1$
$f4(n)=12\ n+ 4$,
$f5(n)=1$,
$f6(n)=5^4 \log n$,
$f7(n)=25\ n^n$,
$f8(n)=4\ n*n$,
$f9(n)=2\log(n^3)+1$

Optional questions
You are not required to submit your answers to optional questions. However, I highly recommend that you attempt them for additional practice.

Question 4) Let $f(n)= n^2 + 2\ n \log n+10^2$. Bound the growth of function f(n) asymptotically. Can you come up with a tight bound? You should clearly justify your answer here (use formal proof).

Question 5) Write a recursive function to implement the binary search algorithm. Analyze its time complexity by formulating a recurrence relation that counts the number of basic operations in the recursive function. Then, solve the recurrence relation to express the function in closed form, resolving all summations and recursive calls. Finally, determine the asymptotic bound for the function in Big-O notation.

Question 6)

A) Count the precise number of "basic operations" executed in the codes below. Your answer should be a function of n ($n \geq 0$) in closed form. Note that "closed form" means that you must resolve all $\sum$'s, recursive relations, etc. An asymptotic answer (such as one that uses Big-O, Theta, and similar) is **not** acceptable. **Show all your work.**

Note: For a recursive function, you first need to write the corresponding recurrence relation and then solve it precisely to come up with the closed form function of n.

```
void fun(int n)
{
```

```
        Perform 1 basic operation;
        if (n >=1)
        {
                fun (n-2);
        }
}
```

B) Express your answer in part A in Big-O. Justify your answer.