Choose any 2 of the 3 problems.
If you answer all three questions, only questions 1 and 2 will be graded.

Question1-Part A) (10 points) For each of the following scenarios, determine the correct relationship between f(n) and g(n). Multiple answers may apply.

Note: Please select the correct options **only**. **Incorrect or extra selections will result in a penalty of -0.5 points per mistake.**

1) Let f(n) and g(n) be two functions representing algorithmic runtimes. The growth rate of f(n) compared to g(n) can be analyzed using the limit below:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$$

where c is a **positive constant**. Based on this limit, identify the correct relationship between f(n) and g(n). Select all correct cases. No justification needed.

a) f(n) = O(g(n)) (Big-O)
b) f(n)=Ω(g(n) (Big-Omega)
c) f(n)=Θ(g(n)) (Theta)
d) f(n)=o(g(n)) (Little-o)
e) f(n)=ω(g(n)) (Little-omega)

2) Now, consider the case where:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Based on this limit, identify the correct relationship between f(n) and g(n). Select all correct cases. No justification needed.

a) f(n) = O(g(n)) (Big-O)
b) f(n)=Ω(g(n) (Big-Omega)
c) f(n)=Θ(g(n)) (Theta)
d) f(n)=o(g(n)) (Little-o)
e) f(n)=ω(g(n)) (Little-omega)

3) Let **f(n)= n!** and **g(n)= n^n** (that is, **n** to the power of **n**). Select all correct cases below. No justification needed.

a) f(n) = O(g(n)) (Big-O)
b) g(n)= O(f(n)) (Big-O)
c) f(n)= Θ(g(n)) (Theta)
d) g(n)= Θ(f(n)) (Theta)
e) f(n)= o(g(n)) (Little-o)
f) g(n)= o(f(n)) (Little-o)

Question1- Part B) (10 points) For each function f(n) below, give **an asymptotic upper bound using "Big-Oh".** You should give the tightest bound possible. No need to justify your answers.

a) $f(n)= 45\ n^2 + \log (n^3)+ n$
b) $f(n)= n^k + 50 \log n + 10$, where $k \geq 1$ is a constant
c) $f(n)= n^2 + n \log n$
d)

$$f(n)=\begin{cases} n^4 + 2n, & n \leq 12 \\ 3n + 5, & n > 12 \ \ and\ n\ is\ odd \\ 12n^2, & n > 12 \ \ and \ \ n\ is\ even \end{cases}$$

Question 2) (20 points)
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 ∑'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.

A)

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

B)

```
for(int i = 1; i <= n; i++)
{
        for (int j = 1; j <= i; j++)
                Perform 1 basic operation;
         //endfor j
 }//endfor i
```

Question 3 (20 points): Set Subtraction Using Singly Linked Lists

We have two sets of integers, **set A** and **set B**, implemented as two **unsorted singly linked lists**. We define **the set difference** operation, **A - B**, as producing a set containing all elements in **A** that are **not present in B**, while maintaining the **original order of elements** in **A**.

For example, given:
A = {1, 2, 8, 4} and **B = {2, 8, 10, 3}**,
the result of **A - B** is **{1, 4}**.

Each linked list node consists of:

- An integer **key**
- A pointer to the **next** node

Assume that **A** has **n** elements and **B** has **m** elements, where **n, m ≥ 0**.

(a) Implementation (15 points)

Write a **C++ function** *subtract* that performs in-place subtraction of two given linked lists, **A** and **B**, as described above. The function should modify **A** directly, removing nodes that contain values found in **B**, while preserving the order of elements in **A**.

Assumptions and Constraints:

- Do **not** create a new or temporary linked list as you implement the *subtract* function.
- The function should return a pointer to the head of the modified list **A**, representing **A - B**.

Example:
If **A** is represented as **1 → 2 → 8 → 4** and **B** as **2 → 8 → 10 → 3**, after subtraction, **A** should be modified to **1 → 4** without allocating extra space for a new list.

Ensure you

(1) **(5 points)** Declare your data structure first.
(2) **(10 points)** Code the search **function** in C++ only as described above.

(b) Complexity Analysis (5 points)

Analyze the **worst-case time complexity** of your function from part (a) and provide a **justification** for your answer.

Note: Your codes should be in C/C++ only. No points for other programming languages.