CMSC 142 Machine Problem 1
Deadline: February 28, 2025 (Friday) @ 7am

Instructions:

1. This will be a **by-pair activity**. You may choose your own pair, and you may also pair up with someone from another section.
2. **Each student must submit the solutions on LMS** with each pair having similar answers on items 1, 4, and 5. However, **items 2-3 must be solved individually**. Each member of the pair must solve either item 2 OR 3 ONLY (e.g. if Nina and Jayv pair up, Nina may choose Item 2 and Jayv may choose Item 3)
3. The submitted file must be in pdf format.

Items:

1. Determine the Big-O complexities of the following snippet of codes. Show the big-O for each block of code (if, for, while, etc.) then evaluate the final big-O from this.

a)
```
void fun(int x) {
    if (x == 0)   return;
    for (int i = 0; i < x; i++)
        op();
    fun(x-1);
}
```

b)
```
void fun(int x) {
    if (x == 0) return;
    for (int i = 0; i < x; i++)
        op();
    for (int i = 0; i < 4; i++)
        fun(x/4);
}
```

c)
```
function fun(int x){
    m = 0
    while m * m < x + 50 do
        op()
        m++
    end while
    return 0
}
```

d) For each of the following pairs of functions f(n) and g(n), determine whether f(n)=O(g(n)), g(n)=O(f(n)), or both. Show your solution/explanation.

   a. $f(n) = \frac{n^2-n}{2}, g(n) = 6n$

   b. $f(n) = n + 2\sqrt{n}, g(n) = n^2$

c. $f(n) = 4nlogn + n, g(n) = \frac{n^2-n}{2}$

2. **(Member 1)** Sort the array A = [64, 57, 13, 70, 85, 39, 22, 48] using
   a. Insertion Sort
   b. Bubble Sort
   c. Selection Sort
   d. Merge Sort
   e. Quick Sort

   Show the step-by-step change in the order until the array is sorted.

3. **(Member 2)** Sort the array A = [57, 22, 70, 13, 85, 48, 64, 39] using
   a. Insertion Sort
   b. Bubble Sort
   c. Selection Sort
   d. Merge Sort
   e. Quick Sort

   Show the step-by-step change in the order until the array is sorted.

4. Summarize the complexities of sorting and searching algorithms in one table:

Algorithm	Best-Case	Scenario / Input Structure for the Best Case	Worst-case	Scenario / Input Structure for the Worst Case
Insertion Sort				
Bubble Sort				
Selection Sort				
Merge Sort				
Quick Sort				
Heap Sort				
Binary Search				
Linear Search				

5. The Bubble Sort algorithm is shown below.

```
1. bubble_sort(array A):
2.     do:
3.         swapped = False
4.     for current = N to 1:
5.         prev = current – 1
6.         if A[prev] > A[current]:
7.             swap A[prev] ↔ A[current]
8.             swapped = True
9.     while swapped = True
```

Show the possible modifications in order to optimize the number of timesteps. Also, explain how the changes results to optimization.

Note: do not replace the whole code with a new one.