Asymptotic Analysis worksheet
CS51 - Fall 2026

Practice Problem 1 - Big O
Using big O notation, simplify the following functions:

1. $3n^3 + 2n + 7$
2. $2^n + n^2$
3. 1000
4. $n + 1000$
5. $1 + \frac{1}{n}$

Practice Problem 2 - Revisiting linear search
Here's the linear search code we have seen before

```
def linear_search(lst, element):
 n = len(lst)
 for i in range(n):
  if lst[i] == element:
   return i
 return -1
```

If we count the number of comparisons (`===`), what is the runtime for `linear_search` in the:

1. worst case:
2. average case:
3. best case:

Use big O notation.

Practice Problem 3 - bubble sort
Here's the bubble sort code we have seen before:

```
def bubblesort (lst):
    n = len(lst)
    for i in range(n):
        for j in range(n-i-1):
            if lst[j]>lst[j+1]:
                exchange(lst, j, j+1)
```

What is the runtime for bubblesort in the worst case?

- For i = 0, we make ?? comparisons:
- For i = 1, we make ?? comparisons:
- For i = 2, we make ?? comparisons:
- ...
- For i = n-2, we make ?? comparisons:
- For i = n-1, we make ?? comparisons:

Adding all these together we get how many comparisons?

Here's the selection sort code we have seen before:

```
def selectionsort (lst):
    n = len(lst)
    for i in range(n-1):
        min_index = i
        for j in range(i+1, n):
            if lst[j]<lst[min_index]:
                min_index = j
        exchange(lst, i, min_index)
```

What is the runtime for selection sort in the worst case?

- For i = 0, we make ?? comparisons:
- For i = 1, we make ?? comparisons:
- For i = 2, we make ?? comparisons:
- ...
- For i = n-2, we make ?? comparisons:

Adding all these together we get how many comparisons?