Homework 3: Analyzing Running Times

Big-$\mathcal{O}$ Notation and Loop Analysis

SCIS 313: Data Structures and Algorithm Analysis
Spelman College

Due: _____

Instructions

For each problem below, you are given a C++ function and asked to determine its **time complexity** in Big-$\mathcal{O}$ notation.

- **Show all work.** Full credit requires a complete written explanation, not just an answer.
- **Fill in the trace table.** Each problem includes a table where you will track loop variables for a small value of n. This helps you see the pattern before generalizing.
- **Write a summation.** Express the total number of iterations as a mathematical sum, then simplify.
- **State your final answer** clearly in Big-$\mathcal{O}$ notation and justify it.

Hints are provided to guide your thinking. Try the problem before reading them.

Problem 1

```
void mystery1(int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i * i; j++) {
            cout << "*";
        }
    }
}
```

Part (a): Trace Table

Complete the table below for $n = 4$. For each value of `i`, write down the upper bound of the inner loop (`i*i`) and count how many times `j` iterates.

Loop Trace for `mystery1` with n

`i`	**Inner loop bound (`i*i`)**	**# of inner iterations**
1		
2		
3		
4		
	Total iterations:	

Part (b): Summation

Express the total number of inner loop iterations as a summation over all values of `i` from 1 to n.

Total $= \sum_{i=1}^{n}$ _____

Part (c): Final Answer

Simplify your summation and state the time complexity. *Recall the formula:* $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

$T(n) = \mathcal{O}($_____$)$

Hint

The inner loop does not run a fixed number of times — its bound depends on `i`. Ask yourself: for a given value of `i`, exactly how many times does the inner loop execute? Write that as a formula in `i`, then sum over all values of `i`.

Problem 2

```
void mystery2(int n) {
    for (int i = n; i >= 1; i /= 2) {
        for (int j = 0; j < n; j++) {
            cout << "*";
        }
    }
}
```

Part (a): Trace Table

Complete the table below for $n = 16$. Track how `i` changes with each outer loop iteration, and note how many times the inner loop runs.

Loop Trace for `mystery2` with n

Outer iter.	`i` (before update)	`i` (after `i /= 2`)	# inner iterations
1	16		
2			
3			
4			
5			
		Total outer iterations:	

Part (b): Count Outer Iterations

How many times does the outer loop execute (in terms of n)? Write a brief explanation using the pattern you observed in the trace table.

Part (c): Final Answer

Using your answer from Part (b), calculate total work and state the time complexity.

$T(n) = \mathcal{O}(\_\_\_\_\_)$

Hint

The inner loop's bound does **not** depend on `i` — look carefully at what it iterates up to. Focus first on counting outer iterations only: if you start at n and keep halving, how many steps before you reach 0?

Problem 3

```
1  void mystery3(int n) {
2      for (int i = 2; i <= n; i++) {
3          for (int j = 2; j <= i; j = j * j) {
4              cout << "*";
5          }
6      }
7  }
```

Part (a): Trace Table

Complete the table below for $n = 20$. For each value of `i`, write out the values `j` takes before the loop exits, and count the iterations.

Loop Trace for `mystery3` with n

`i`	Values of `j`: 2, ...	# of inner iterations
2		
3		
4		
5		
10		
16		
17		
20		

Part (b): Inner Loop Analysis

For a fixed value of `i`, the inner loop starts at $j = 2$ and updates $j \leftarrow j^2$. Write out the sequence of values j takes: 2, 4, 16, 256, ...

After k iterations, what is the value of j in terms of k? (Hint: notice the pattern $2^1, 2^2, 2^4, 2^8, \ldots$)

After k iterations: $j =$ _____

The loop stops when $j > i$. Using your formula, how many iterations k happen before the loop stops? Express your answer in terms of $\log(\log(i))$.

# of inner iterations for a given $i = \mathcal{O}($_____$)$

Part (c): Final Answer

State the overall time complexity.

$T(n) = \mathcal{O}($_____$)$

Hint

The update `j = j * j` is very aggressive — much faster than doubling. After just a few iterations, `j` gets enormous. Write out the first few values of `j` explicitly for $i = 20$ and count how many steps it takes. Then ask: how does this count grow as i grows?

Problem 4

```
1  void mystery4(int n, int m) {
2      for (int i = 0; i < n; i++) {
3          cout << "*";
4      }
5      for (int j = 0; j < m; j++) {
6          cout << "*";
7      }
8  }
```

Part (a): Trace Table

Complete the table below for $n = 4,\ m = 6$.

Loop Trace for `mystery4` with n

Loop	Variable and bound	# of iterations
First loop	`i` from 0 to $n-1$	
Second loop	`j` from 0 to $m-1$	
	Total:	

Part (b): Combining Independent Terms

The two loops are sequential (not nested). Explain in 1–2 sentences why you **add** the two counts rather than multiply them.

Part (c): Can We Simplify Further?

Consider the following two scenarios:

(i) **Scenario A:** We know nothing about the relationship between n and m.
What is $T(n, m)$?

$T(n, m) = \mathcal{O}($_____$)$

(ii) **Scenario B:** We are told that $m = \mathcal{O}(n)$ (i.e., m grows no faster than n).
Can you simplify? What is $T(n)$ in this case?

$T(n) = \mathcal{O}(\_\_\_\_\_)$

Hint

Both n and m are independent inputs. Unless you are given information about how they relate to each other, you cannot drop either variable from your answer. Think about what would happen if $n = 10$ and $m = 1{,}000{,}000$.

Problem 5

```
1  void mystery5(int n) {
2      for (int i = 0; i < n; i++) {
3          for (int j = i; j < n; j++) {
4              cout << "*";
5          }
6      }
7  }
```

Part (a): Trace Table

Complete the table below for $n = 5$. For each value of `i`, note what value `j` starts at and how many times the inner loop runs.

Loop Trace for `mystery5` with n

`i`	`j` starts at	`j` ends at	# of inner iterations
0			
1			
2			
3			
4			
		Total iterations:	

Part (b): Summation

For a given value of `i`, the inner loop runs from `j = i` to `j = n-1`, which is $n - i$ iterations. Write the total number of iterations as a summation:

$$\text{Total} = \sum_{i=0}^{n-1} (n - i) = \_\_\_\_\_$$

Hint: You may find it helpful to reindex. Let $k = n - i$, so as i goes from 0 to $n - 1$, k goes from n down to 1.

Part (c): Final Answer

Simplify and state the time complexity.

$T(n) = \mathcal{O}($_____$)$

Hint

At first glance this looks like $\mathcal{O}(n^2)$ just because there are two nested loops. That happens to be correct — but make sure you can *prove* it by writing the sum explicitly. The key observation is that the inner loop runs a *different* number of times for each `i`; your table should make this very clear.