Construct a mathematical function modeling the worst-case runtime of the following method. Your function should be written in terms of n, the provided input.

Assume each println takes some constant time c to run.

```java
public void mystery(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n * n; j++) {
            System.out.println("Hello"); +1
        }
        for (int j = 0; j < 10; j++) {
            System.out.println("world"); +1
        }
    }
}
```

Remember: work outside in

Solution: \( T(n) = n(n^2 + 10) = n^3 + 10n \)
Project 1 Out

Project 1 out – Part 1 due Friday April 13th
- Pair Programming
- Navigating the Project
- How it will be graded
- Need a partner? Come see me after class or email me ASAP

HW 1 due tonight!
- please make sure your legal name is associated with your practice-it account

Class Survey due tonight
Modeling Complex Loops

```java
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!");
    }
}
```

Keep an eye on loop bounds!
Modeling Complex Loops

for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!");  
    }
}

Summation
1 + 2 + 3 + 4 +... + n = \sum_{i=1}^{n} i

Definition: Summation

\sum_{i=a}^{b} f(i) = f(a) + f(a + 1) + f(a + 2) + ... + f(b-2) + f(b-1) + f(b)

T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} c
Simplifying Summations

for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println("Hello!");
    }
}

\[ T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{i-1} c \]

= \sum_{i=0}^{n-1} ci \quad \text{Summation of a constant}

= c \sum_{i=0}^{n-1} i \quad \text{Factoring out a constant}

= c \frac{n(n-1)}{2} \quad \text{Gauss's Identity}

= \frac{c}{2}n^2 - \frac{c}{2}n \quad \text{O(n^2)}
Function Modeling: Recursion

```java
public int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
```
Function Modeling: Recursion

```java
public int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
```

\[
T(n) = \begin{cases} 
    C_1 & \text{when } n = 0 \text{ or } 1 \\
    C_2 + T(n-1) & \text{otherwise}
\end{cases}
\]

**Definition: Recurrence**

Mathematical equivalent of an if/else statement

\[f(n) = \]
Unfolding Method

\[ T(n) = \begin{cases} 
C_1 & \text{when } n = 0 \text{ or } 1 \\
C_2 + T(n-1) & \text{otherwise}
\end{cases} \]

\[ T(3) = C_2 + T(3 - 1) = C_2 + (C_2 + T(2 - 1)) = C_2 + (C_2 + (C_1)) = 2C_2 + C_1 \]

\[ T(n) = C_1 + \sum_{i=0}^{n-1} C_2 \]

Summation of a constant

\[ T(n) = C_1 + (n-1)C_2 \]
Asymptotic Analysis
Asymptotic Analysis

A function $f(n)$ is **dominated** by $g(n)$ when...

There exists two constants $c > 0$ and $n_0 > 0$

Such that for all values of $n \geq n_0$

$f(n) \leq c \times g(n)$

**Definition: Domination**

Can we say that $n$ “dominates” $4n$?

Yes!

$c = 4$ or more

$n_0 = 1$ (because definition requires $n_0 > 0$)
Asymptotic Analysis

Clearly $n^2$ dominates $n$, right?

We can find a $c$ & $n_0$ that satisfy the definition of domination:

$$c = 1$$
$$n_0 = 1$$

Definition: Big O

$O(f(n))$ is the “family” or “set” of all functions that are dominated by $f(n)$.
∈ Element Of

f(n) is less than or equal to g(n)

“f(n) is dominated by g(n)"

“f(n) is contained inside O(g(n))”

f(n) ∈ O(g(n))
**Practice**

- $5n + 3 \in O(n)$: True
- $n \in O(5n + 3)$: True
- $5n + 3 = O(n)$: True
- $O(5n + 3) = O(n)$: True
- $O(n^2) = O(n)$: False
- $n^2 \in O(1)$: False
- $n^2 \in O(n)$: False
- $n^2 \in O(n^2)$: True
- $n^2 \in O(n^3)$: True
- $n^2 \in O(n^{100})$: True

**Definition: \( \in \)**

- "element of" mathematical symbol

**Definition: Big O**

- $O(f(n))$ is the “family” or “set” of all functions that are dominated by $f(n)$
### Definitions: Big $\Omega$

**Definition: Big $\Omega$**

$\Omega(f(n))$ is the "family" or "set" of all functions that are dominated by $f(n)$
- $\Omega(f(n))$ is less than or equal to $f(n)$
- $\Omega(f(n)) \leq f(n)$
- $\Omega(f(n))$ is the lower bound of $f(n)$'s asymptotic analysis

Do we have a name for a set of functions that are the dominators?

**Definition: Big $\Omega$**

$\Omega(f(n))$ is the family of all functions that dominate $f(n)$
- $\Omega(f(n))$ is greater than or equal to $f(n)$
- $\Omega(f(n)) \geq f(n)$
- $\Omega(f(n))$ is the upper bound of $f(n)$'s asymptotic analysis
Examples

\[ 4n^2 \in \Omega(1) \]
true

\[ 4n^2 \in \Omega(n) \]
true

\[ 4n^2 \in \Omega(n^2) \]
true

\[ 4n^2 \in \Omega(n^3) \]
false

\[ 4n^2 \in \Omega(n^4) \]
false

\[ 4n^2 \in O(1) \]
false

\[ 4n^2 \in O(n) \]
false

\[ 4n^2 \in O(n^2) \]
true

\[ 4n^2 \in O(n^3) \]
true

\[ 4n^2 \in O(n^4) \]
true

---

**Definition: Big O**

\( O(f(n)) \) is the “family” or “set” of all functions that are dominated by \( f(n) \)

**Definition: Big \( \Omega \)**

\( \Omega(f(n)) \) is the family of all functions that dominates \( f(n) \)
**Definitions: Big $\Theta$**

We say $f(n) \in \Theta(g(n))$ when both

$f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$ are true

Which is only when $f(n) = g(n)$

**Definition: Big $\Theta$**

$\Theta(f(n))$ is the family of functions that are equivalent to $f(n)$

Industry uses “Big $\Theta$” and “Big $O$” interchangeably
Summary

\[ O(f(n)) \leq f(n) = \Theta(f(n)) \leq \Omega(f(n)) \]

<table>
<thead>
<tr>
<th>f(n)</th>
<th>Dominated by</th>
<th>Dominates</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(1) )</td>
<td>( f(n) \in O(g(n)) )</td>
<td></td>
</tr>
<tr>
<td>( O(\log n) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( O(n) )</td>
<td></td>
<td>( f(n) \in \Omega(g(n)) )</td>
</tr>
<tr>
<td>( O(n^2) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( O(n^3) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>