Title

Solution to Problem 1: Loop Invariant
Computing Binomial Coefficient Using Pseudocode and Loop Invariant

---

1. Original Pseudocode

```
get_binomial(n, k):
1   x ← 1
2   for i ← 1 to k do
3       x ← x · (n − i + 1)
4       x ← x / i
5   return x
```

---

2. Binomial Coefficient Formula

The binomial coefficient counts the number of ways to choose$k$elements from$n$elements without repetition:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

---

a. Loop Invariant

Invariant: At the beginning of each iteration$i$,

$$x = \frac{\prod_{j=1}^{i-1}(n-j+1)}{(i-1)!}$$

Interpretation: After $i-1$ iterations, x equals the partial binomial coefficient:

$$x = \frac{n \cdot (n-1) \cdots (n-i+2)}{1 \cdot 2 \cdots (i-1)}$$

---

b. Proof of Loop Invariant

- **Base Case:** Before the loop starts, $x = 1, i = 1$Formula gives:

$$\frac{\prod_{j=1}^{0}(n-j+1)}{0!} = 1$$

Invariant holds.

- **Inductive Step:** Assume invariant holds at start of iteration i:

$$x = \frac{\prod_{j=1}^{i-1}(n-j+1)}{(i-1)!}$$

After lines 3 and 4:

$$x \leftarrow x \cdot (n-i+1), \quad x \leftarrow x/i$$

So:

$$x = \frac{\prod_{j=1}^{i}(n-j+1)}{i!}$$

Invariant holds for next iteration.

---

c. Correctness

After k iterations:

$$x = \frac{\prod_{j=1}^{k}(n-j+1)}{k!} = \frac{n(n-1)\cdots(n-k+1)}{k!} = \binom{n}{k}$$

Thus, the algorithm returns the correct binomial coefficient.

---

d. Python One-Liners

Using `math.comb`:

```
import math
binomial = lambda n, k: math.comb(n, k)
```

Without `math.comb`:

```
from functools import reduce
binomial = lambda n, k: reduce(lambda a,b: a*b, range(n-k+1, n+1)) //
reduce(lambda a,b: a*b, range(1, k+1))
```

---

extra. Complexity & Efficiency

- **Time Complexity:** $OK)$because the loop runs$k$times.
- **Space Complexity:** $O(1)$since only a few variables are used.
- This algorithm avoids computing full factorials, making it efficient for large n and small k.