Insertion Sort

At the beginning of each iteration of the **for** loop, which is indexed by j, the elements $A[1 \ldots j-1]$ are the elements *originally* in positions 1 through $j-1$, but now in sorted order. We state these properties of $A[1 \ldots j]$ formally as a ***loop invariant***:

> At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1 \ldots j-1]$ consists of the elements originally in $A[1 \ldots j-1]$, but in sorted order.

We use this loop invariant to help us understand why this algorithm is correct. We must show three things about the loop invariant:

> **Initialization:** It is true prior to the first iteration of the loop.
>
> **Maintenance:** If it is true before an iteration of the loop, it remains true before the next iteration.
>
> **Termination:** When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Fill the Best Case and Worst Case columns with appropriate values from the given choices.

INSERTION-SORT(A)	*cost*	**Best Case**	**Worst Case**
1 **for** $j = 2$ **to** $A.length$	c_1		
2 $key = A[j]$	c_2		
3 // Insert $A[j]$ into the sorted sequence $A[1 \mathinner{.\,.} j-1]$.	0		
4 $i = j - 1$	c_4		
5 **while** $i > 0$ and $A[i] > key$	c_5		
6 $A[i+1] = A[i]$	c_6		
7 $i = i - 1$	c_7		
8 $A[i+1] = key$	c_8		

Choices:

a. n

b. $n-1$

c. $\frac{n(n-1)}{2}$

d. $\frac{n(n+1)}{2} - 1$

e. 0