Given a list `sequence` and a predicate `P`, return the index of the first element for which the predicate holds, or -1 if none exists. Formally, return

$$\begin{array}{lll} -1 & \text{if } \forall\ j \in [0, n), \sim P(\texttt{sequence}[j]) & \\ & & \\ k & \text{otherwise, where} & P(\texttt{sequence}[k]) \\ & \hfill\text{and} & \forall\ j \in [0, k), \sim P(\texttt{sequence}[j]) \end{array}$$

Given a list `sequence` and a predicate `P`, return the index of the first element for which the predicate holds, or -1 if none exists. Formally, return

$$\begin{array}{lll} -1 & \text{if } \forall\ i \in [0, n), \sim P(\mathtt{sequence}[i]) & \\ & & \\ k & \text{otherwise, where} & P(\mathtt{sequence}[k]) \\ & \qquad\qquad\qquad \text{and} & \forall\ i \in [0, k), \sim P(\mathtt{sequence}[i]) \end{array}$$

Invariant 1 (Loop of `bounded_linear_search`.)

(a) $\forall\ j \in [0, \mathtt{i} - 1), \sim P(\mathtt{sequence}[j])$

(b) `found` *iff* $P(\mathtt{sequence}[\mathtt{i} - 1])$

(c) `i` *is the number of iterations completed.*

(a) $\forall\, j \in [0, \texttt{i} - 1), \sim P(\texttt{sequence}[j])$

(b) $\texttt{found}$ iff $P(\texttt{sequence}[\texttt{i} - 1])$

(c) $\texttt{i}$ is the number of iterations completed.

Initialization.

(a) Since $\texttt{i}$ is initially 0, the range $[0, \texttt{i}) = [0, 0)$ which is empty. Hence the proposition is vacuously true.

(b) With $\texttt{i} = 0$, $\texttt{sequence}[\texttt{i} - 1]$ doesn't exist. However, it's reasonable to interpret $P(\mathit{undef})$ as false, which makes this part of the invariant hold.

(c) There have been 0 iterations, and $\texttt{i} = 0$.

(a) $\forall\ j \in [0, \mathtt{i} - 1), \sim P(\mathtt{sequence}[j])$

(b) `found` iff $P(\mathtt{sequence}[\mathtt{i} - 1])$

(c) `i` is the number of iterations completed.

Maintenance. Since the variable `i` itself changes during the execution of an iteration, we distinguish between its value when the iteration starts from its value when the iteration finishes by i_{pre} and i_{post}, respectively. Note that $i_{\text{post}} = i_{\text{pre}} + 1$. Similarly distinguish $\mathtt{found}_{\text{pre}}$ and $\mathtt{found}_{\text{post}}$

(a) It must be that $\sim \mathtt{found}_{\text{pre}}$ or else the guard would have failed and the loop would have terminated before this iteration. Thus $\sim P(\mathtt{sequence}[i_{\text{pre}} - 1])$, by the *inductive hypothesis*, part b. Together with the fact that that
$\forall\ j \in [0, i_{\text{pre}} - 1), \sim P(\mathtt{sequence}[j])$, we now have
$\forall\ j \in [0, i_{\text{pre}}), \sim P(\mathtt{sequence}[j])$, that is $\forall\ j \in [0, i_{\text{post}} - 1), \sim P(\mathtt{sequence}[j])$.

(b) Immediate from the assignment to `found`.

(c) Immediate from the update to i. □

Correctness Claim 1 (`bounded_linear_search.`)

After at most n iterations, `bounded_linear_search` will return as specified.

Proof. By Invariant 1.c, after at most n iterations, $\mathtt{i} = n$ and the guard will fail. Moreover, when the guard fails, either `found` or $\mathtt{i} = n$. Consider the cases of `found` and $\sim$ `found`.
Case 1. Suppose `found`. Then we return $i - 1$. Invariant 1.a tells us that nothing in $[0, \mathtt{i} - 1)$ satisfies P. Invariant 1.b tells us that $\mathtt{i} - 1$ does. Together these fulfill the second part of the specification: $\mathtt{i} - 1$ is the first item satisfying P, and we return it.
Case 2. Suppose $\sim$ `found`. By elimination $\mathtt{i} = n$. Invariant 1.a tells us that nothing in $[0, n - 1)$ satisfies P. Invariant 1.b tells us that $\mathtt{i} - 1$, that is, $n - 1$, also does not satisfy P. We return -1, fulfilling the first part of the specification. □

Given a list sequence sorted by a given total order TO and given an item, return

$$\begin{array}{ll} -1 & \text{if } \forall\ i \in [0, n), \texttt{sequence}[i] \neq \texttt{item} \\ k & \text{otherwise, where } \texttt{sequence}[k] = \texttt{item} \end{array}$$

Invariant 3 (Loop of binary_search.)

(a) *If* $\exists\ j \in [0, n)$ *such that* $\texttt{item} = \texttt{sequence}[j]$*, then* $\exists\ j \in [\texttt{low}, \texttt{high})$ *such that* $\texttt{item} = \texttt{sequence}[j]$.

(b) *After* i *iterations,* $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

(a) If $\exists\ j \in [0, n)$ such that $\texttt{item} = \texttt{sequence}[j]$,
then $\exists\ j \in [\texttt{low}, \texttt{high})$ such that $\texttt{item} = \texttt{sequence}[j]$.

(b) After i iterations, $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

Initialization.

(a) Initially $\texttt{low} = 0$ and $\texttt{high} = n$, so the hypothesis and conclusion are identical.

(b) No iterations yet, so

$$\texttt{high} - \texttt{low} = n - 0 = n = \frac{n}{1} = \frac{n}{2^0}$$

(a) If $\exists\ j \in [0, n)$ such that $\texttt{item} = \texttt{sequence}[j]$,
then $\exists\ j \in [\texttt{low}, \texttt{high})$ such that $\texttt{item} = \texttt{sequence}[j]$.

(b) After i iterations, $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

Maintenance. Distinguish $\texttt{low}_{\text{pre}}$ and $\texttt{low}_{\text{post}}$, $\texttt{high}_{\text{pre}}$ and $\texttt{high}_{\text{post}}$. Let i be the number of iterations completed. We're given that if $\exists\ j \in [0, n)$ such that $\texttt{item} = \texttt{sequence}[j]$, then $\exists\ j \in [\texttt{low}_{\text{pre}}, \texttt{high}_{\text{pre}})$ such that $\texttt{item} = \texttt{sequence}[j]$; also that $\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}} \leq \frac{n}{2^{i-1}}$ (this is our *inductive hypothesis*). The guard also assures us that $\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}} > 1$.

We have three possibilities, corresponding to the if-elif-else:

(a) If $\exists\ j \in [0, n)$ such that $\texttt{item} = \texttt{sequence}[j]$, then $\exists\ j \in [\texttt{low}, \texttt{high})$ such that $\texttt{item} = \texttt{sequence}[j]$.

(b) After i iterations, $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

Case 1: Suppose $\texttt{item} < \texttt{sequence}[\texttt{mid}]$.

(a) Since $\texttt{sequence}$ is sorted, $\forall\ j \in [\texttt{mid}, \texttt{high}_{\text{pre}})$, $\texttt{item} < \texttt{sequence}[j]$. Thus if $\exists\ j \in [\texttt{low}_{\text{pre}}, \texttt{high}_{\text{pre}})$, then $\exists\ j \in [\texttt{low}_{\text{pre}}, \texttt{mid})$, that is (with the update to $\texttt{high}$ but not to $\texttt{low}$), $\exists\ j \in [\texttt{low}_{\text{post}}, \texttt{high}_{\text{post}})$
Now, by transitivity of the conditional, if $\exists\ j \in [0, n)$ such that $\texttt{item} = \texttt{sequence}[j]$, then $\exists\ j \in [\texttt{low}_{\text{post}}, \texttt{high}_{\text{post}})$ such that $\texttt{item} = \texttt{sequence}[j]$.

(b) If the length of the range is odd, then the sub-ranges above and below mid are of equal size, each half of the range length minus one. If the range length is even, then the lower subrange is half that size and the upper subrange is one less than half. Either way we throw away at least half and keep no more than half. So,

$$\texttt{high}_{\text{post}} - \texttt{low}_{\text{post}} \leq \frac{1}{2} \cdot (\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}}) \leq \frac{1}{2} \cdot \frac{n}{2^{i-1}} \leq \frac{n}{2^i}$$

(a) If $\exists\, j \in [0, n)$ such that $\texttt{item} = \texttt{sequence}[j]$, then $\exists\, j \in [\texttt{low}, \texttt{high})$ such that $\texttt{item} = \texttt{sequence}[j]$.

(b) After i iterations, $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

Case 2: Suppose $\texttt{item} = \texttt{sequence}[\texttt{mid}]$.

(a) Immediately we have $\exists\, j \in [\texttt{mid}, \texttt{mid} + 1)$, and, with the update to $\texttt{high}$ and $\texttt{low}$, that means $\exists\, j \in [\texttt{low}_{\text{post}}, \texttt{high}_{\text{post}})$. Moreover, the conditional is $T \rightarrow T \equiv T$.

(b) Note $\texttt{high}_{\text{post}} - \texttt{low}_{\text{post}} = 1$. Earlier we said $1 < \texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}} \leq \frac{n}{2^{i-1}}$. Since $\texttt{high}_{\text{pre}} - \texttt{low}_{\text{pre}}$ must be a whole number, $2 \leq \frac{n}{2^{i-1}}$, and so $1 \leq \frac{n}{2^i}$. Finally $\texttt{high}_{\text{post}} - \texttt{low}_{\text{post}} \leq \frac{n}{2^i}$.

Case 3: Suppose $\texttt{item} > \texttt{sequence}[\texttt{mid}]$. This is similar to Case 1. □

Correctness Claim 3 (`binary_search`.)

After at most $\lg\ n$ iterations, `binary_search` returns as specified.

Proof. Suppose $i \geq \lg\ n$. Then $2^i \geq n$ and $\frac{n}{2^i} \leq 1$. Hence $\texttt{high} - \texttt{low} \leq 1$ and the guard fails.
Invariant 3.a still means that if the item is anywhere, it's in the range. The guard implies that on loop exit the range has size 0 or 1.
Suppose the range has size 0. Then the item isn't in the range (since nothing is), and thus it isn't anywhere. Since $\texttt{high} = \texttt{low}$, the first part of the conditional fails and and -1 is returned, as specified.
On the other hand, suppose the range has size 1. We still don't know if the item is in the range, but we have only one location to check. If it's in `sequence[low]`, then we return `low`, which meets the specification. Otherwise the second part of the condition fails and -1 is returned, as specified. □

Invariant 3 (Loop of `binary_search`.)

(a) *If* $\exists\ j \in [0, n)$ *such that* $\texttt{item} = \texttt{sequence}[j]$, *then* $\exists\ j \in [\texttt{low}, \texttt{high})$ *such that* $\texttt{item} = \texttt{sequence}[j]$.

(b) *After* i *iterations,* $\texttt{high} - \texttt{low} \leq \frac{n}{2^i}$.

Invariant 4 (Preconditions of `binary_search_recursive`)

(a) *If* $\exists\ j \in [0, n)$ *such that* $\texttt{item} = \texttt{sequence}[j]$, *then* $\exists\ j \in [\texttt{start}, \texttt{stop})$ *such that* $\texttt{item} = \texttt{sequence}[j]$.

(b) $\texttt{start} \leq \texttt{stop}$

Invariant 5 (Outer loop of `selection_sort`)

(a) *The range $[0, i)$ in `sequence` is sorted.*

(b) *All the elements in range $[0, i)$ in `sequence` are less than or equal to all the elements in the range $[i, n)$.*

(c) *i is the number of iterations completed.*

Invariant 6 (Inner loop of `selection_sort`)

(a) $\texttt{sequence}[\texttt{min\_pos}] = \texttt{min}$.

(b) *`min` is the smallest element in the range $[i, j)$. (Formally: $\forall\ k \in [i, j), \texttt{min} \leq \texttt{sequence}[k]$.)*

(c) *$j - i - 1$ is the number of iterations completed.*

Correctness Claim 4 (`selection_sort`)

After n iterations, `sequence` is sorted and `selection_sort` returns.