Sequential search in an array

Prove the following program correct w.r.t. the given specification. The program sequentially search for the element 'x' in array 'a'.

```
{n >= 0}
begin
      i := 1;
      found := 0;
      while i <= n do
            if x = a[i] then
                  found := 1;
                  ind := i;  fi;
            i := i + 1;
      od
end
{(exists j (1 <= j <= n and a[j] = x) => found != 0 and a[ind] = x
and 1 <= ind <= n) and  (forall j (1 <= j <= n => a[j] != x) =>
found = 0)}
```

Before we start, notice that the content of the array is never modified in the program (i.e. there are no assignments where an array cell is the left-hand-side part). Therefore we don't have to give any special treatment to the instructions involving the array.

Choice of loop invariant

Basically, after iteration i, all the elements before index i in a satisfy the postcondition with i substituted for n. Furthermore, we must express the fact that the index i never exceeds n + 1. Therefore:

I = `{(exists j (1 <= j < i and a[j] = x) => found != 0 and a[ind] = x and 1 <= ind < i) and  (forall j (1 <= j < i => a[j] != x) => found = 0) and 1 <= i <= n + 1}`

Proof substeps

As usual, we split the proof into three substeps:

```
1. {Pre} i := 1; found := 0; {I}
2. {I} while ... od {I and i > n}
3. {I and i > n} => {Post}
```

Proof of step 1

A simple double backsubstitution proves step 1.

```
{(exists j (1 <= j < 1 and a[j] = x) => 0 != 0 and a[ind] = x and 1
<= ind < 1) and 1 <= 1 <= n + 1} =(the antecedent of the first
implication is false because the inequality 1 <= j < 1 cannot hold)=
{1 <= n + 1} == {n >= 0} == {Pre}

i := 1;

{(exists j (1 <= j < i and a[j] = x) => 0 != 0 and a[ind] = x and 1
<= ind < i) and  (forall j (1 <= j < i => a[j] != x) => 0 = 0) and 1
<= i <= n + 1}

found := 0;

{(exists j (1 <= j < i and a[j] = x) => found != 0 and a[ind] = x
and 1 <= ind < i) and  (forall j (1 <= j < i => a[j] != x) => found
= 0) and 1 <= i <= n + 1}
```

Proof of step 3

```
{(exists j (1 <= j < i and a[j] = x) => found != 0 and a[ind] = x
and 1 <= ind < i)  and  (forall j (1 <= j < i => a[j] != x) => found
= 0) and 1 <= i <= n + 1 and i > n}
```

Notice that `1 <= i <= n + 1` and `i > n` imply `i = n+1`. Therefore we get:

```
{(exists j (1 <= j < n+1 and a[j] = x) => found != 0 and a[ind] = x
and 1 <= ind < n+1)  and  (forall j (1 <= j < n+1 => a[j] != x) =>
found = 0) and i = n + 1} == {(exists j (1 <= j <= n and a[j] = x)
=> found != 0 and a[ind] = x and 1 <= ind <= n)  and  (forall j (1
<= j <= n => a[j] != x) => found = 0) and i = n + 1} =>
```
(a fortiori)
```
{Post}
```

Proof of step 2

By IR4, step 2 is equivalent to:

```
{I and i <= n} if .. fi; i := i + 1; {I}
```

We backsubstitute once throught the last assignment:

```
{I and i <= n}

if ... fi;

{(exists j (1 <= j <= i and a[j] = x) => found != 0 and a[ind] = x
and 1 <= ind <= i) and  (forall j (1 <= j <= i => a[j] != x) =>
found = 0) and 0 <= i <= n} =def= {Q}

i := i + 1;

{I}
```

Now, we can apply IR3b, thus reducing to proving:

```
1.{I and i <= n and x = a[i]} found := 1; ind := i; {Q}
2.{I and i <= n and x != a[i]} => {Q}
```

Step 2.1 is proved by two backsubstitutions of {Q}:

```
{(exists j (1 <= j <= i and a[j] = x) => 1 != 0 and a[i] = x and 1
<= i <= i)  and  (forall j (1 <= j <= i => a[j] != x) => 1 = 0) and
0 <= i <= n} == {(exists j (1 <= j <= i and a[j] = x) => a[i] = x
and 1 <= i) and not forall j (1 <= j <= i => a[j] != x) and 0 <= i
<= n} == {(exists j (1 <= j <= i and a[j] = x) => a[i] = x and i
>=1) and exists j(1 <= j <= i and a[j] = x) and 0 <= i <= n} == {a
[i] = x and i >=1) and 0 <= i <= n} =def= {R}

found := 1;

{(exists j (1 <= j <= i and a[j] = x) => found != 0 and a[i] = x and
1 <= i <= i)  and  (forall j (1 <= j <= i => a[j] != x) => found =
0) and 0 <= i <= n}

ind := i;

{Q} =def= {(exists j (1 <= j <= i and a[j] = x) => found != 0 and a
[ind] = x and 1 <= ind <= i)  and  (forall j (1 <= j <= i => a[j] !=
x) => found = 0) and 0 <= i <= n}
```

Now, notice that:

```
{I and i <= n and x = a[i]} == {(exists j (1 <= j < i and a[j] = x)
=> found != 0 and a[ind] = x and 1 <= ind < i)  and  (forall j (1 <=
j < i => a[j] != x) => found = 0) and 1 <= i <= n and x = a[i]}
```

Trivially `1 <= i <= n  =>  0 <= i <= n and i >= 1`.

Furthermore, `x = a[i]` so, it is true that `exists j (1 <= j <= i and a[j] = x)`.

This concludes the proof of step 2.1.

Now, for step 2.2.

```
{I and i <= n and x != a[i]} == {(exists j (1 <= j < i and a[j] = x)
=> found != 0 and a[ind] = x and 1 <= ind < i) and  (forall j (1 <=
j < i => a[j] != x) => found = 0) and 1 <= i <= n + 1 and i <= n and
x != a[i]} == {(exists j (1 <= j < i and a[j] = x) => found != 0 and
a[ind] = x and 1 <= ind < i) and  (forall j (1 <= j < i => a[j] !=
x) => found = 0) and 1 <= i <= n and x != a[i]}
```

Now, if `exists j(1 <= j < i and a[j] = x)`, then `found != 0, a[ind] = x and 1 <= ind < i`, so the first term of the conjunction in Q is subsumed a fortiori. The second term is also implied, since if `exists j(1 <= j < i and a[j] = x)`, then the antecedent of the implication `(forall j (1 <= j <= i => a[j] != x) => found = 0)` is false.

If, instead, `forall j(1 <= j < i => a[j] != x)`, then `found = 0 and forall j(1 <= j <= i => a[j] !=x)`, since it is also true that `x != a[i]`. So, the second term of the conjunction in Q is subsumed. The first term is also implied, since if `forall j(1 <= j <= i => a[j] !=x)` then it is false that `(exists j (1 <= j <= i and a[j] = x))`, rendering the first implication in Q identically true.

Finally, just notice that `1 <= i <= n  =>  0 <= i <= n`.

This concludes step 2.2 and, in turn, the whole partial correctness proof.

Array inversion

Precondition: $\{n >= 0\}$

Postcondition: $\{\forall\ i\ ((1 \le i \le n) \rightarrow b[i]=a[n-i+1]\}$

Note 1: n = 0 means that the array is empty. In this case array b is of course the inverse of a.

Note 2: the two arrays are supposed to be of equal length (this is a static property that needn't be proved).

```
begin
  h:= 1;
  while h <= n do
    b[h]:=a[n-h+1];
    h:=h+1;
  od
end
```

Select a loop invariant

$I = \{\ \forall\ i\ ((1\le i<h) \rightarrow b[i]=a[n-i+1]) \wedge\ h \le n+1\ \}$

According to the composition rule IR1, we can split the proof in three steps:

```
1. {n >= 0} h:= 1; {I}
2. {I} while . . . od {I and h>n}
3. {I and h>n}  → Postcondition
```

Proof of point 1

By trivially applying backward substitution, we get:

$\{\ \forall\ i\ ((1\le i <1) \rightarrow b[i]=a[n-i+1]) \wedge\ 1 \le n+1\ \} = \{\ \forall\ i\ (\text{false} \rightarrow ...) \wedge\ 0 \le n\ \} = \{0 \le n\}$

which is exactly the precondition.

Proof of point 2

According to rule IR4 we have to prove that:

$\{I \wedge h \le n\ \}$

```
b[h]:=a[n-h+1];
h:=h+1;
```

$\{I\}$

By backward substitution through h:=h+1, we get:

$\{I \wedge h \le n\ \}$

```
b[h]:=a[n-h+1];
```

$\{\ \forall\ i\ ((1\le i < h+1) \rightarrow b[i]=a[n-i+1]) \wedge\ h+1 \le n+1\ \}$ =(adapting the indices)=

$\{\ \forall\ i\ ((1\le i \le h) \rightarrow b[i]=a[n-i+1]) \wedge\ h \le n\ \}$

By backward substitution through `b[h]:=a[n-h+1]`, we get:

$I^* = \{\ \forall\ i\ ((1\le i \le h) \rightarrow \{\text{if } i=h \text{ then } a[n-i+1]=a[n-i+1] \text{ else } b[i]=a[n-i+1]\}) \wedge\ h \le n\ \}$

```
b[h]:=a[n-h+1];
```

$\{ \forall i ((1 \le i \le h) \rightarrow b[i]=a[n-i+1]) \wedge h \le n \}$

When i = h the right part of the implication is identically true, thus we can rewrite I* as follows:

$I^* = \{ \forall i ((1 \le i < h) \rightarrow b[i]=a[n-i+1]) \wedge h \le n \}$

$I \wedge h \le n = \{ \forall i ((1 \le i < h) \rightarrow b[i]=a[n-i+1]) \wedge h \le n+1 \wedge h \le n\} == I^*$

Proof of point 3

$\{I \wedge h>n\} == \{ \forall i ((1 \le i < h) \rightarrow b[i]=a[n-i+1]) \wedge h \le n+1 \wedge h>n \} ==$

$\{ \forall i ((1 \le i < h) \rightarrow b[i]=a[n-i+1]) \wedge h = n+1 \} ==$

$\{ \forall i ((1 \le i < n+1) \rightarrow b[i]=a[n-i+1]) \wedge h = n+1 \} ==$

$\{ \forall i ((1 \le i \le n) \rightarrow b[i]=a[n-i+1]) \wedge h = n+1 \}$

The first conjunct is exactly the Postcondition.

Ex. 5.1.13, pg. 227 exercisebook (Bubblesort)

```
begin
  i:=n;
  while i>=1 do
    j := 1;
    while j<i do
      k := j+1;
      if a[j] > a[k]
        then x:=a[k]; a[k]:=a[j]; a[j]:=x;
      fi
      j := j+1;
    od
    i = i-1;
  od
end.
```

Definition of pre- and post- conditions

A dummy array b is used to guarantee that at the end of the computation array a contains exactly the same elements as before starting the computation. Notice that we want this to be true *through the whole* computation.

We also suppose that all the values contained in 'a' are distinct. (This must also hold *through the whole* computation)

So, all in all, we have:

Precondition: $\{n\geq 0 \wedge \text{permutation}(a,b) \wedge \text{distinct}(a)\}$

Postcondition: $\{\text{permutation}(a,b) \wedge \text{ordered}(a) \wedge \text{distinct}(a)\}$

Note: n = 0 means that the array is empty. In this case there is nothing to do.

permutation(a,b) $\equiv\{\forall\, i\ ((1\leq i \leq n) \rightarrow \exists j\ 1\leq j \leq n \wedge a[i]=b[j])\}$

distinct(a) $\equiv \{\forall i \forall j\ ((1\leq i \leq n \wedge 1\leq j \leq n \wedge a[i]=a[j]) \rightarrow i=j)\}$

a[x:y] indicates the portion of 'a' having index in the interval [x,y], including the extremes (as an aside, a syntax similar to this is used in Python).

ord(a,i) $\equiv \{\forall\, h\ ((i+1\leq h < n) \rightarrow a[h]<a[h+1]\}$ // a[i+1:n] is ordered

ordered(a) $\equiv \{\forall\, i\ ((1\leq i < n) \rightarrow a[i]<a[i+1]\}$

gr(a,i) $\equiv \{\forall\, h\ ((i+1\leq h \leq n) \rightarrow \forall\, m\ ((1\leq m \leq i) \rightarrow a[m]<a[h]))\}$

// elements in a[i+1:n] are greater that elements in a [1:i]

max(a,j) $\equiv \{\forall\, h\ ((1\leq h < j) \rightarrow a[j]>a[h]\}$ // a[j] is greater than any element in a[1:j-1]

As a consequence of the definitions above, it is:

ordered(a) $\rightarrow$ ord(a,m) $\forall\, m\ (0\leq m \leq n)$

ordered(a) = ord(a,0)

true = ord(a,n) = gr(a,n) = max(a,1) (because the acceptable ranges are empty)

According to the composition rule IR1, we can split the proof in three steps:

```
1. {Precondition} i:=n; {I}
```

`2. {I} while . . . od {I ∧ i < 1}` (note: this is the external while loop)
`3. {I ∧ i < 1} → Postcondition`

Select external loop invariant

In the external loop we have that a[i+1:n] is ordered. Moreover, a[i+1] is greater than any element in a[1:i]. i is always between 0 and n. Thus, the invariant can be defined as follows:

$I = \{permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge n \geq i \geq 0 \wedge distinct(a)\}$

Proof of point 1

Applying backward substitution of I through i:=n,

$\{permutation(a,b) \wedge ord(a,n) \wedge gr(a,n) \wedge n \geq n \geq 0 \wedge distinct(a)\} = \{permutation(a,b) \wedge n \geq 0 \wedge distinct(a)\}$

which is equal to the precondition.

Proof of point 2

According to the composition rule IR1, and while-loop-rule IR4, we can split the proof into three steps:

`2.1.{I ∧ i ≥ 1 } j:=1; {J}`
`2.2.{J} while . . . od {J ∧ j ≥ i}` (note: this is the internal while loop)
`2.3.{J ∧ j ≥ i } i:=i-1; {I}`

Select internal loop invariant

The external loop invariant is valid, since a[i+1:n] is not modified.

At every iteration a[j] is greater than any element in a[1:j-1]

$J = \{ I \wedge max(a,j) \wedge 1 \leq j \leq i \wedge i \geq 1\}$

Proof of point 2.1

By backward substitution we get (notice that I does not contain variable j):

$\{I \wedge max(a,1) \wedge 1 \leq i \wedge 1 \leq 1 \leq i \} == \{I \wedge 1 \leq i\}$

, which is exactly what we had to find.

Proof of point 2.2

This step is, as usual, reduced by IR4 to:

$\{J \wedge j < i\}$

```
k := j+1;
if a[j] > a[k]
  then x:=a[k]; a[k]:=a[j]; a[j]:=x;
fi
j := j+1;
```

$\{J\}$

We can play a little "trick" on the first assignement. In fact, since the value of k which is used is *always* j+1, we can avoid backsubstitution and immediately plug this "alias" in the local precondition. In other words we reduce to:

$\{J \wedge j< i \ \wedge k = j+1\}$ // when used k is always equal to j+1

```
if a[j] > a[k]
  then x:=a[k]; a[k]:=a[j]; a[j]:=x;
fi
j := j+1;
```

$\{J\}$

<u>Proof of the "else" case (by IR3)</u>

$\{J \wedge \ j< i \ \wedge k = j+1 \wedge a[j] \leq a[k]\}$ j := j+1; $\{J\}$

$\{J \wedge j< i \ \wedge k = j+1 \wedge a[j] \leq a[k]\}$ =

$\{ I \ \wedge \max(a,j) \wedge 1 \leq j \leq i \wedge i\geq 1 \wedge j< i \ \wedge k = j+1 \wedge a[j] \leq a[j+1]\}$ =

$\{ I \ \wedge \max(a,j) \wedge i\geq 1 \wedge 1 \leq j < i \ \wedge k = j+1 \wedge a[j] \leq a[j+1]\} = I\#$

By backwardsubstitution of J through j:=j+1 we get

$\{ I \ \wedge \max(a,j+1) \wedge 0 \leq j < i \wedge i \geq 1\} = J^*$

J* is implied by I#, since $\{distinct(a) \wedge \max(a,j) \wedge a[j] \leq a[j+1]\} \rightarrow \max(a,j+1)$

<u>Proof of the "then" case (by IR3):</u>

$\{J \wedge \ j< i \ \wedge k = j+1 \wedge a[j] > a[k]\} = \{J \wedge \ j< i \ \wedge k = j+1 \wedge a[j] > a[j+1]\}$=

$\{ I \ \wedge \max(a,j) \wedge 1 \leq j < i \wedge k = j+1 \wedge a[j] > a[j+1]\}$ =

$\{n \geq i \geq 0 \wedge distinct(a) \wedge permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge i\geq 0 \wedge \max(a,j) \wedge \ 1 \leq j < i \wedge k = j+1 \wedge a[j] > a[j+1]\}$ =

$\{n \geq i \geq 0 \wedge distinct(a) \wedge permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge \max(a,j) \wedge \ 1 \leq j < i \wedge k = j+1 \wedge a[j] > a[j+1]\}$

```
x:=a[k]; a[k]:=a[j]; a[j]:=x;
```

$\{J^*\} = \ \{ I \ \wedge \max(a,j+1) \wedge 0 \leq j < i \wedge i \geq 1\}$ =

$\{\{n \geq i \geq 0 \wedge distinct(a) \wedge permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge \max(a,j+1) \wedge 0 \leq j < i \wedge i \geq 1\}$

<u>Let us consider the effect of the array swap</u>

The effect of x:=a[k]; a[k]:=a[j]; a[j]:=x; can be represented as follows (by backward substitution on 'a' elements, introducing a new dummy variable 'h'):

a[h] = (if h=j then a[k] else (if h = k then a[j] else a[h]))

```
x:=a[k];  // here the array is on the rhs of the assignement
```

a[h] = (if h=j then x else (if h = k then a[j] else a[h]))

```
a[k]:=a[j];  // change value a[h]
```

a[h] = (if h=j then x else a[h])

```
a[j]:=x;
```

Let a#[h] = (if h=j then a[k] else (if h = k then a[j] else a[h])) be the effect of this triple backwardsubstitution on the array value a[h]. Generalizing, for any predicate f, let f# be the effect of the triple backwardsubstitution on f.

Now, we can backsubstitute J* by observing separately that:

permutation#(a,b) = $\{\forall$ h $((1\leq h \leq n) \rightarrow \exists m\ 1\leq m \leq n \wedge a[h]=b[m])\}\#$ = $\{\forall$ h $((1\leq h \leq n) \rightarrow \exists m\ 1\leq m \leq n \wedge b[m] =$ (if h=j then a[k] else (if h = k then a[j] else a[h]))$\}$

Now, we show that: {permutation(a,b) $\wedge$ k=j+1} $\rightarrow$ permutation#(a,b)

In fact: if h = j: $\{\exists m\ 1\leq m \leq n \wedge b[m] = a[k]\}$ follows from permutation(a,b) by considering h = k and the fact that $1\leq j+1 \leq n$ (since $j < i \leq n$).

If h = k: $\{\exists m\ 1\leq m \leq n \wedge b[m] = a[j]\}$ follows from permutation(a,b) by considering h = j and the fact that $j < i \leq n$.

Finally, if h != k and h != j, then the implication follows trivially.

Now, we show that {distinct(a)$\wedge$ k = j+1} $\rightarrow$ distinct#(a).

distinct(a) = $\{\forall h \forall m\ ((1\leq h \leq n \wedge 1\leq m \leq n \wedge a[h]=a[m]) \rightarrow h=m)\}$

distinct#(a) = $\{\forall h \forall m\ ((1\leq h \leq n \wedge 1\leq m \leq n \wedge$

(if h = j then a[k] else if h = k then a[j] else a[h]) =

(if m = j then a[k] else if m = k then a[j] else a[m]))$\rightarrow$ h=m}

This is proved by considering exaustively all the cases.

1. h=j $\wedge$ m = j: a[k] = a[k] $\rightarrow$ h=m (true because h=m)
2. h=j $\wedge$ m = k: a[k] = a[j] $\rightarrow$ h=m (true because a[k]=a[j+1]!=a[j] because distinct (a))
3. h=j $\wedge$ m != j,k: a[k] = a[m] $\rightarrow$ h=m (true because k!=m, so a[k]!=a[m] because distinct(a))
4. h=k $\wedge$ m = j: a[j] = a[k] $\rightarrow$ h=m (true because a[k]=a[j+1]!=a[j] because distinct (a))
5. h=k $\wedge$ m = k: a[j] = a[j] $\rightarrow$ h=m (true because h=m)
6. h=k $\wedge$ m != j,k: a[j] = a[m] $\rightarrow$ h=m (true because j!=m, so a[j]!=a[m] because distinct(a))
7. h!=j,k $\wedge$ m = j: a[h] = a[k] $\rightarrow$ h=m (true because k!=h, so a[k]!=a[h] because distinct(a))
8. h!=j,k $\wedge$ m = k: a[h] = a[j] $\rightarrow$ h=m (true because j!=h so a[h]!=a[j] because distinct (a))
9. h!=j,k $\wedge$ m != j,k: a[h] = a[m] $\rightarrow$ h=m (true because distinct(a))

It is also simple to note that:

{ord(a,i) $\wedge$ j< i} = $\{\forall$ h $((i+1\leq h<n) \rightarrow a[h]<a[h+1]) \wedge j< i\ \}$ $\rightarrow$ ord#(a,i) = $\{\forall$ h $((i+1\leq h<n) \rightarrow a\#[h]<a\#[h+1]) \wedge j< i\ \}$

, since the ordering of the portion of the array a[i+1, n] is not affected; in fact the elements involved in the change are at most up to position j + 1, and j < i (or j + 1 <= i).

$\{gr(a,i) \wedge j< i\} \rightarrow gr\#(a,i) = \{\forall h\ ((i+1 \le h \le n) \rightarrow \forall m\ ((1 \le m \le i) \rightarrow a[m]<a[h]))\}$

, because of the same reason as the ord() predicate (see right above).

$\{max(a,j) \wedge k = j+1 \wedge a[j] > a[j+1]\} \rightarrow max\#(a,j+1)$

, since after swapping a[j] with the next element (which is initially smaller), a[j+1] becomes greater than the elements in a[1:j].

All in all, we have proved that:

$\{n \ge i \ge 0 \wedge distinct(a) \wedge permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge max(a,j) \wedge 1 \le j < i \wedge k = j+1 \wedge a[j] > a[j+1]\} \rightarrow$

$\{J^*\} = \{I \wedge max(a,j+1) \wedge 0 \le j < i \wedge i \ge 1\} = \{\{n \ge i \ge 0 \wedge distinct(a) \wedge permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge max(a,j+1) \wedge 0 \le j < i \wedge i \ge 1\}$

, which concludes step 2.2.

Proof of point 2.3

$\{J \wedge j \ge i\} = \{I \wedge max(a,j) \wedge j \le i \wedge i \ge 1 \wedge j \ge i \ge 1\} =$

$\{I \wedge max(a,j) \wedge j=i \wedge i \ge 1\} = \{I \wedge max(a,i) \wedge j=i \wedge i \ge 1\} \equiv I'$

By backward substitution of I through i:=i-1; we get

$\{permutation(a,b) \wedge ord(a,i-1) \wedge gr(a,i-1) \wedge i-1 \ge 0\} =$

$\{permutation(a,b) \wedge ord(a,i-1) \wedge gr(a,i-1) \wedge i \ge 1\} =$

$\{permutation(a,b) \wedge ord(a,i) \wedge a[i]>a[i-1] \wedge gr(a,i-1) \wedge i \ge 1\} =$

$\{permutation(a,b) \wedge ord(a,i) \wedge a[i]>a[i-1] \wedge gr(a,i) \wedge max(a,i) \wedge i \ge 1\} =$

$\{I \wedge max(a,i) \wedge i \ge 1\} \equiv I'$, since

$I = \{permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge i \ge 1\}$

and $max(a,i) \rightarrow a[i] > a[i-1]$, so point 2.3 is proved.

Proof of point 3

$\{I \wedge i < 1\} = \{permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge n \ge i \ge 0 \wedge distinct(a) \wedge i < 1\} =$

$\{permutation(a,b) \wedge ord(a,i) \wedge gr(a,i) \wedge n \ge i = 0 \wedge distinct(a)\} =$

$\{permutation(a,b) \wedge ord(a,0) \wedge gr(a,0) \wedge n \ge i = 0 \wedge distinct(a)\} =$

$\{permutation(a,b) \wedge ordered(a) \wedge gr(a,0) \wedge n \ge i = 0 \wedge distinct(a)\}$

$\rightarrow$ Postcondition