Correctness of Algorithm

An algorithm is correct if it satisfies its specifications
Specifications are often written using preconditions and postconditions

Precondition: a statement involving the variable used in the algorithm. It says that certain facts must be true before an execution of the algorithm begins, it can describe the allowable input

Postcondition: a statement about the variables used in the algorithm, it says that certain facts must be true when an execution of the algorithm ends, it often describes the correct outputs for a given input

Partially correct: the algorithm is partially correct IFF

if i) The precondition holds, ii) the algorithm is executed, and iii) it eventually halts then the postcondition holds

Total correctness: the algorithm is partially correct and has termination

Example

```
z ← 0
w ← y
while w ≠ 0 do
   z ← z + w
   w ← w − 1
```

Precondition: $y \in \mathbb{N}$, postcondition: $z = \frac{(1+y)y}{2}$
If $y < 0$, the algorithm doesn't halt

Specifications for Search$(A\text{: Array}, k\text{: key})$

Preconditions: A is an array and k has the same type as elements in A

Postconditions: Return $i \in \mathbb{Z}^+ .\ i \le \text{len}(A)$ and $A[i] = k$, if such i exists, otherwise return 0

And A, k are not changed by the program

Specification for BinarySearch$(A\text{: Array}, k\text{: key})$

Precondition: A is sorted in non-decreasing order

Postcondition: same as Search

Specification for Sort$(A\text{: Array})$

Precondition: elements in A are from a totally ordered domain

Postcondition: the multiset of elements in A remains unchanged,

And the elements are in non-decreasing order $\left(i \le i \le j \le \text{len}(A) \text{IMPLIES } A[i] \le A[j]\right)$

Specification for Merge$(A\text{: Array}, B\text{: Array})$

Precondition: A, B are sorted in non-decreasing order and are from the same totally ordered domain

Postcondition: The multiset of elements in C is equal to the union of the multisets of elements in A and B

And the elements of C are in non-decreasing order.

Example

MergeSort$(A\text{: Array}, n\text{: int})$

```
if n > 1, then
   m ← ⌈n/2⌉
   U ← A[1, m]
   V ← A[m + 1, n]
   MergeSort(U, m)
```

6. $\quad$ MergeSort$(V, n - m)$
7. $\quad A \leftarrow$ Merge(U, V)

Proof $\quad$ For $n \in \mathbb{N}$, let $P(n) \coloneqq$ for all arrays A, $A[1 \dots n]$ with elements have a totally ordered domain, if MergeSort(A, n) is performed, then it eventually halts at which time A is sorted in non-drecreasing order and the multiset of elements in A is unchanged
Let $n \in \mathbb{N}$ be arbitrary, let $A[1 \dots n]$ be an arbitrary array of elements from a toally ordered domain
Assume $\forall m \in \mathbb{N}. \left(m < n \text{ IMPLIES } P(m)\right)$

Suppose $n = 0{,}1$, then the test on line 1 fails, algorithm halts and A is unchanged, A is vacuously in non-decreasing order

By generalization, $P(0), P(1)$

Suppose $n > 1$, then the test on line 1 succeeds, $m = \lceil n/2 \rceil$, so $0 \le m, m - n < n$
By induction hypothesis, after MergeSort(U, m) and MergeSort$(V, n - m)$, U, V are soted in non-decreasing order, the multiset of elements in U is the multiset of elements of $A[1 \dots m]$ and the multiset of elements in V is the multiset of elements of $A[m + 1 \dots n]$
It follows from the specification of Merge(U, V) thata after $A \leftarrow$ Merge(U, V) is performed, the elements in A are sorted in non-decreasing order, and the multiset of A is the union of the multiset of elements in U and V
Then, the multiset of A is unchanged

By generalization and strong induction, $\forall n \in \mathbb{N}. P(n)$

Example

QuickSort(A)

1. if length$(A) > 1$ then
2. $\quad p \leftarrow A[1]$
3. $\quad$ partition A into
   $L \leftarrow$ multiset of elements in A that are less than p
   $E \leftarrow$ multiset of elements in A that are equal to p
   $G \leftarrow$ multiset of elements in A that are greater than p
4. $\quad$ QuickSort(L)
5. $\quad$ QuickSort(G)
6. $\quad A \leftarrow L + E + G$

Proof $\quad$ For all $n \in \mathbb{N}$, let $P(n) \coloneqq$ for all arrays A, $A[1 \dots n]$ with elements have a totally ordered domain, if QuickSort(A, n) is performed, then it eventually halts at which time A is sorted in non-drecreasing order and the multiset of elements in A is unchanged
Let $n \in \mathbb{N}$ be arbitrary, let $A[1 \dots n]$ be an arbitrary array of elements from a toally ordered domain
Assume $\forall m \in \mathbb{N}. \left(m < n \text{ IMPLIES } P(m)\right)$

Suppose $n = 0{,}1$, then the test on line 1 fails, algorithm halts and A is unchanged, A is vacuously in non-decreasing order

By generalization, $P(0), P(1)$

Suppose $n > 1$, then the test on line 1 succeeds, $p = A[1]$
By line 3, $\forall l \in L. \forall e \in E. \forall g \in G. l \le e \le g$
Since $A[1] \in E, |L| < n, |G| < n$, by induction hypothsis, L, G are each sorted in non-decreasing order and multiset of L, G are unchanged
By line 6, the multiset of elements in A is the union of the multiset of L, E, G, which is A's partitions, which the multiset is unchanged

By generalization and strong induction, $\forall n \in \mathbb{N}. P(n)$