1st June 1971

Dear Niklaus,

I took immediate note of your unfavourable reaction to my theory of parallel programming, and sat down to rethink it again. I hope you like the new version better. I think that there are fewer ideas and notations involved, and certainly the proof principles are quite simple, But it is still very difficult to construct convincing examples of the use of parallelism.

I am proposing to travel from Zurich to Munich by air on the 6.10 p.m. flight on Sunday, 18th July. I certainly hope we can travel together.

My lectures at Marktoberdorf will not contain much that is new. I propose to describe the proof methods which have been discovered so far, including procedures, parameters, jumps, recursion, parallelism and perhaps incorporating new material on coroutines. I too propose to base my notations on PASCAL, and hope to have a draft of a formal definition by the time we meet.

Yours sincerely,

(C.A.R.Hoare)

Professor Niklaus Wirth,
Zurich.

Eidg. Technische Hochschule Zürich
Fachgruppe Computer-Wissenschaften

Clausiusstrasse 55
CH-8006 Zürich

Prof. C.A.R. Hoare
Computer Science Dept.
University of Belfast

BELFAST BT7 1NN
Northern Ireland

April 27, 1971

Dear Tony:

Many thanks for your letter and enclosures. My idea about a joint paper on axiomatisation of Pascal came out of the feeling that it simply ought to be done, and better now than later. However the dilemma is that I don't know how to find the time. I am now teaching the large (350 students) course on programming, using Pascal for the first time. I have the ambition to write detailed lecture notes, hopefully to be publishable. And this is no little task, as I begin to realise.

The "axiomatic paper" should anyway be short, concentrating on what we know how to do it. The axioms must truly reflect the language, be consistent, but not necessarily complete. I would certainly omit go to statements, classes, and possibly even variant records.

Could you possibly come to Zürich before or after the Marktoberdorf affair for a few days? (Our semester ends July 16, and I am taking off to Stanford around August 15).

I read your paper on parallel processes with interest. It left me a bit bewildered, and not quite satisfied. I feel that it should be possible to succeed with fewer new ideas and notations.

I enclose two examples of verifications which I tried to work out. Doesn't it turn out in practice, that one automatically uses the "weakest necessary antecedent". The Pascal notation led to

$\{P\}\ Q\ \{R\}$ instead of $P\ \{Q\}\ R$.

Finally, I enclose a copy of a letter concerning bootstrapping techniques. Please let me know when you intend to send Jim Welsh over. During June, we will have a Jim Wells here with the same purpose (U. of Winnipeg).

Yours sincerely,

Nklaus

Prof. N. Wirth

```
{ Binary search }
var i,j,k: integer;
{ A_m < A_{m+1} < ... < A_n, m≤n }
i := m;  j := n;
repeat {(i≤j) ∧ (A_{>i}<x) ∧ (A_{>j}>x) ∧ ¬f }
   k := (i+j) div 2;
   { i≤k≤j }
   if A_k = x then
      begin f := true
         { f ∧ (A_k=x) }
      end
   else {(A_k≠x) ∧¬f}
   if A_k < x then
      begin {A_{≤k} < x}
         i := k+1
         { A_{<i} < x }
      end else
      begin {A_k>x ⟶ A_{≤k}>x}
         j := k-1
         { A_{>j}>x }
      end
 {((A_k=x) ∧ f) ∨ ((A_k≠x) ∧ ¬f ∧ (A_{<i}<x) ∧ (A_{>j}>x))}
until i > j;
if f then {A_k=x} else {(A_{<i}<x) ∧ (A_{>j}>x) ∧ (i>j)}
                       ⟶ ¬ ∃ A_t=x
```

```
{ Binary search }
var i,j,k: integer;
{ A_m<A_{m+1}< ... <A_n, m≤n }
i := m;  j := n;
repeat { (i≤j) ∧ (A_{<i}<x) ∧ (A_{>j}>x) }
   k := (i+j) div 2;
   { i≤k≤j }
   if A_k ≤ x then
      begin {(A_k ≤ x) ∧ (A_{<k} > x)}
         i := k+1
         {(k=i-1) ∧ ((A_k=x) ∨ (A_{<i}<x))}
      end
   else {(A_k>x) ∧ (A_{<i}<x)} ;
   { ((A_k<x) ∧ (A_{<i}<x) ∨ ((A_k=x) ∧ (k=i-1)) ∨ (A_k>x) ∧ (A_{<i}<x))}
   if A_k ≥ x then
      begin {(A_k≤x) ∧ (A_{>k}>x)}
         j := k-1
         { (k=j+1) ∧ ((A_k=x) ∨ (A_{>j}>x))}
      end
   else {(A_k<x) ∧ (A_{>j}>x)} ;
{((A_k=x) ∧ (k=i-1=j+1)) ∨ ((A_k≠x) ∧ (A_{<i}<x) ∧ (A_{>j}>x))}
until i > j
{(A_j=x) ∨ ((A_{≤i}<x) ∧ (A_{>j}>x) ∧ (i>j))}
{(A_k=x) ∨ (¬∃A_t=x)}
```