On a somewhat disappointing correspondence

I did not react to Frank Rubin's letter [0], confident that all my potential comments would be made by others, but in the five letters published two months later [1], I found none of them expressed. So, I reluctantly concluded that I had better record my concerns, big and small.

(0) The problem statement refers to an N by N matrix X ; Rubin's programs refer to an n by n matrix x . In other contexts this might be considered a minor discrepancy, but I thought that by now professional programmers had learned to be more demanding on themselves and not to belittle the virtue of accuracy. I shall stick to the capital letters.

(1) Rubin still starts indexing the rows and the columns at 1 . I thought that by now professional programmers knew how much more preferable it is to let the natural numbers start at 0 . I shall start indexing at 0 .

(2) Rubin's third program fails for N = 0 (in which case his second program succeeds only by accident −see below−). I thought that by now professional programmers would know the stuff the silly bugs are made of.

(3) Rubin's second program fails to detect the first all-zero row if it is the last row of the

matrix.

(4) Rubin's third program relies -without stating it explicitly- on the "conditional <u>and</u>", for which, if the first operand is <u>false</u>, the second operand is allowed to be undefined. The conditional connectives - "<u>cand</u>" and "<u>cor</u>" for short - are, however, less innocent than they might seem at first sight. For instance, <u>cor</u> does not distribute over <u>cand</u>: compare

$$(A \ \underline{\text{cand}}\ B)\ \underline{\text{cor}}\ C \quad \text{with} \quad (A\ \underline{\text{cor}}\ C)\ \underline{\text{cand}}\ (B\ \underline{\text{cor}}\ C) \ ;$$

in the case $\neg A \wedge C$, the second expression requires B to be defined, the first one does not. Because the conditional connectives thus complicate the formal reasoning about programs, they are better avoided.

(5) Rubin's letter effectively conceals that his problem can be solved systematically by a nested application of the same algorithm (sometimes known as "The bounded linear search"). His statement of the problem is: "Let X be an $N \times N$ matrix of integers. Write a program that will print the number of the first all-zero row of X, if any." Now, concentrate to begin with on the "if any"; nothing should be printed if all rows differ from the all-zero row, formally, if

$$(\underline{A} i: 0 \leq i < N: \neg(\underline{A} j: 0 \leq j < N: X[i,j] = 0)) \quad .$$

The theorem of "The bounded linear search" states for any boolean function B on the first N natural numbers ($N \geq 0$)

```
|[ var f: bool ; var n: int
; f, n := true, 0 {P}
; do f ∧ n ≠ N → f, n := B(n), n+1 od
  {P ∧ (¬f ∨ n = N)}
]|
```

with the invariant P given by

$$P: \quad 0 \leq n \leq N \quad \wedge$$
$$(f \equiv (\underline{A}k : 0 \leq k < n : B(k))) \quad \wedge$$
$$(\underline{A}k : 0 \leq k < n-1 : B(k)) \quad ,$$

which states that, upon termination
in the case f : all B's are true , and
in the case ¬f : n-1 is the smallest value for which B is false .

Applying the above theorem twice yields for Rubin's problem

```
|[ var c: bool; var i : int
; c, i := true, 0
; do c ∧ i ≠ N →
      |[ var d : bool ; var j : int
      ; d, j := true, 0
      ; do d ∧ j ≠ N → d, j := X[i,j]=0 , j+1 od
      ; c, i := ¬d, i+1
      ]|
  od
; if c → skip [] ¬c → print (i-1) fi
]|
```
,

and for me this settles the problem.

By my standards, a competent professional programmer in 1987
(i) should recognize that Rubin's problem asks to be solved by a nested application of the same algorithm;
(ii) should know the theorem of "The bounded linear search";
(iii) should be able to derive that theorem and its proof;
(iv) should not hesitate to use it;
(v) should not waste his time in pointing out that the boolean variable d is superfluous;
(vi) should keep his repetitions simple and disentangled
(vii) etc. .

Evidently, my priorities are not shared by every one, for Rubin's letter and most of the five reactions it evoked were conducted instead in terms of all sorts of "programming language features" that seem better ignored than exploited. The whole correspondence was carried out at a level that vividly reminded me of the intellectual climate of twenty years ago, as if stagnation were the major characteristic of the computing profession, and that was a disappointment.

References

[0] Rubin, Frank ""GOTO Considered Harmful" Considered Harmful" Commun ACM 30, 3 (Mar 1987), 195-196

[1] Moore, Donald and Musciano, Chuck and Liebhaber, Michael J. and Lott, Steven F. and Starr, Lee

""'"GOTO Considered Harmful" Considered Harmful" Considered Harmful?". Commun. ACM 30, 5 (May 1987), 351-355

Pasadena, 25 May 1987

prof. dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
United States of America