"Bulterman's reshuffling problem", by

W.H.J. Feijen,

Department of Mathematics,

Technological University of Eindhoven,

P.O. Box 513,

5600 MB Eindhoven,

The Netherlands.

Bulterman's reshuffling problem

(The sole purpose of this exercise is to give a development and a presentation of the solution of a programming problem, with some more emphasis on the presentation. At the end of the text, when a program is shown, the reader should be convinced of the correctness of the design.
Handles on the notations and techniques used can be obtained from [0a] or [0b] and from [1], if so desired.)

Consider an array $X(0 .. N-1)$, such that

- $N = R * M$, $R \geq 1$ and $M \geq 1$
- R of its elements are red
- the remaining $N - R$ elements are blue .

It is requested to permute the array elements such that

(a) the relative order of the blue elements remains unaltered, and
(b) the red elements are moved to the positions $r * M$, $0 \leq r < R$.

*

The standard trick to trivialize the proof that the final value of X is a permutation of its original value is to stick to array alterations of the form X: swap(i,j) , $0 \leq i \leq j < N$. So, let us adopt the trick.

Of the two remaining requirements (a) and (b) , the first one seems to be the hardest: incautiously swapping of blue elements is likely to create irrevocable chaos. So, let us try to swap the blue elements with care:

- swapping two distinct blue elements is so harmful to (a) that we simply disallow it;
- swapping a red and a blue element is at least suspect, unless all elements "in between" are red, in which case we (had better) allow it.

Thus, having covered all cases, we arrive at a regime in which swapping two elements is constrained to the two outer elements of a non-empty "train of elements", all elements of which are red with the possible exception of one element at an end, which may be blue.

More specifically, if we see to it that X: swap(i,j) takes place subject to the condition

Tr(i,j): $(\underline{A}h\colon i \le h < j\colon \ X(h) \text{ is red })$,

then (a) is "automatically" satisfied.

When obeying this rule for swapping elements our remaining obligation consists of fulfilling (b) , i.e. of establishing

Z: $(\underline{A}k\colon 0 \le k < R\colon \ X(k * M) \text{ is red })$,

which mentions red elements only.
We can view {Tr(i,j)} X: swap(i,j) as a movement of the red train (X(i),..., X(j-1)) over one position "to the right".
So, comes the idea that the problem of satisfying Z is readily solved if we succeed in forming a red train comprising all red elements at the left end of the array X and then let it travel to the right, each time uncoupling its left-most wagon at the appropriate positions.
If we can move a red train to the right we can move it to the left as well, and if we can uncouple a wagon we can also couple one, simply by inverting the operations. So, the problem of allocating a red train comprising all red elements at the left end of the array X can be solved by letting a red train travel from right to left picking up all red wagons it encounters.
More precisely, we propose

(0) {Tr(i,j)} X: swap(i,j); i, j := i + 1, j + 1 {Tr(i,j)}
as the mechanism to move the train to the right, and hence

(1) {Tr(i,j)} i, j := i - 1, j - 1; X: swap(i,j) {Tr(i,j)}
as the mechanism to move it to the left;

(2) {Tr(i,j) <u>and</u> i < j} i:= i + 1 {(X(i-1) is red) <u>and</u> Tr(i,j)}
as the procedure to uncouple a wagon at the left end of the train, and hence

(3) {Tr(i,j) <u>and</u> (X(i-1) is red)} i:= i - 1 {i < j <u>and</u> Tr(i,j)}
as the procedure to couple a wagon at the left end.

With the abbreviations

P(r): $0 \le r \le R$ <u>and</u> $(\underline{A}k\colon 0 \le k < r\colon \ X(k * M) \text{ is red})$
Q(j): $(\underline{A}h\colon j \le h < N\colon \ X(h) \text{ is blue})$,

the program -- which is somewhat lavishly annotated -- is:

```
begin i, j, r: int
   ; i, j := N, N
     {Tr(i,j)} {Q(j)} {0 ≤ i ≤ j ≤ N}
   ; do i ≠ 0 →
          {i > 0}
          if X(i-1) is red → i:= i - 1 {see (3)}
          [] X(i-1) is blue → i, j := i - 1, j - 1; X: swap(i,j) {see (1)}
          fi
          {Tr(i,j)} {Q(j)} {0 ≤ i ≤ j ≤ N}
     od
     {Tr(i,j)  and  Q(j)  and  i = 0, hence  j = R}
   ; r:= 0
     {Tr(i,j)} {P(r)} {j = R - r + i} {i ≤ r * M}
   ; do r ≠ R →
          {r < R}
          {Tr(i,j)} {j = R - r + i} {i ≤ r * M}
          do i ≠ r * M →
               {i < r * M , hence  j < N}
               X: swap(i,j); i, j := i + 1, j + 1
               {Tr(i,j), see (0)} {j = R - r + i} {i ≤ r * M}
          od
          {Tr(i,j)  and  j = R - r + i  and  r < R, hence  Tr(i,j)  and  i < j,
          hence (X(i) is red)} {i = r * M} {P(r)}
        ; i, r := i + 1, r + 1
          {Tr(i,j), see (2)} {P(r)} {j = R - r + i} {i ≤ r * M}
     od
     {P(r)  and  r = R,  hence  Z}
end.
```

*

Acknowledgment: R.W. Bulterman is acknowledged for having posed the problem. (End of Acknowledgment)

[0a] Dijkstra, Edsger W., "A Discipline of Programming", Prentice-Hall, 1976.

[0b] Dijkstra, Edsger W., "Guarded Commands, Nondeterminacy and Formal Derivation of Programs", Comm. ACM 18, 453-457 (1975).

[1] Hoare, C.A.R., "An Axiomatic Basis for Computer Programming", Comm. ACM 12, 576-580 (1969) .