An examination exercise, designed by W.H.J.Feijen.

The problem posed to the students that had followed last fall's course "An Introduction to the Art of Programming" boiled down to the following:

"Given an integer value N $(N \geq 3)$ and a monotonically increasing sequence of integer values $X(0),\ldots,X(N)$ with $X(0) = 0$. For $0 \leq i < N$, the value $X(i)$ represents the clockwise distance along a circle with circumference $X(N)$ of the point P_i from the point P_0 . Given also an integer value K $(K \geq 3)$. Design a program determining whether among the points P_0 through P_{N-1} , K different points can be selected that are the vertices of a regular K-gon. Because the answer is clearly "no" if $X(N)$ mod $K \neq 0$, $X(N)$ mod $K = 0$ may be assumed."

* * *

Let $d = X(N)/K$: d is then the distance between successive vertices of such a K-gon. We propose to inspect the P_i in the order of increasing i . Our answer is "yes" if in the so-called "last segment" --i.e. consisting of the points with a distance y from P_0 that satisfies $X(N) - d \leq y < X(N)$ -- a point can be found such that its $K-1$ "predecessors" --i.e. points at distances d, $2d$, $3d$, ... smaller-- all occur. That is, our answer can be expressed as

$$(E\ y: X(N) - d \leq y < X(N): v(y)) \quad ,$$

in which the predicate $v(y)$ is defined for $0 \leq y < X(N)$ by

$$v(y) = (E\ i: 0 \leq i < N: X(i) = y) \text{ and } (0 \leq y < d \text{ or } v(y-d)) \quad ,$$

a recursive definition that is suggested by the above "all" . The first term expresses that y must be the distance of one of the given points, the second term expresses that all its "predecessors" must be present as well.

The idea of the algorithm is to keep track of a set

$$s = \{\ y \mid b \leq y < b + d \text{ and } v(y)\}$$

initialized for $b = 0$ --where the term $0 \leq y < d$ settles the membership-- and to increase b until the range from b to $b + d$ covers all the points in the last segment for which $v(y)$ holds. The final answer is then established by

yes:= non empty(s) .

To begin with, b = the smallest element in s ; we propose to maintain that relation as long as s is not empty. Because the size of s is a monotically non-increasing function of b , empty(s) for $b < X(N) - d$ implies empty(s) for $b = X(N) - d$.

With b defined for non empty(s) as the smallest element of s , a first sketch of our program is

```
d:= X(N)/ K; s:= empty set; i:= 0;
do X(i) < d → add X(i) to s ; i:= i + 1 od;
do non empty(s) cand X(N) > b + d →
     q:= (E i: 0 ≤ i < N: X(i) = b + d);
     if q → replace in s the value b by b + d
      [] non q → remove from s the value b
     fi
od;
yes:= non empty(s).
```

When we represent the set s by the array ms, the elements of which are the members of s in the order of increasing magnitude, the value b , when defined, equals ms.low .

The assignment to q can be achieved by

$$q := (X(i) = b + d)$$

provided i is the smallest solution of $X(i) \geq b + d$. Introducing the proposed representation for s and eliminating b and q we get the somewhat more "official" program

```
begin glocon N, X, K; vircon yes; pricon d; privar ms, i;
      d vir int:= X(N)/ K; ms vir int array := (0); i vir int := 0;
      do X(i) < d → ms:hiext(X(i)); i:= i + 1 od;
      do ms.dom > 0 cand X(N) > ms.low + d →
            i:= "the smallest solution of  X(i) ≥ ms.low + d";
            if X(i) = ms.low + d → ms:lorem; ms:hiext(X(i))
             [] X(i) > ms.low + d → ms:lorem
            fi
      od;
      yes vir bool := ms.dom > 0
end
```

Note that the guard "$X(N) > ms.low + d$" guarantees the existence of an i satisfying $X(i) \geq ms.low + d$. The minimal solution of $X(i) \geq ms.low + d$ can be found by a linear search, the invariant relation of which

$$(\underline{A}\ j:\ 0 \leq j < i:\ X(i) < ms.low + d)$$

is not destroyed by the alternative construct and can, therefore, be taken outside the repetitive construct. Note that this relation already holds after the first repetitive construct, where we have $ms.low = 0$. The assignment

i:= "the smallest solution of $X(i) \geq ms.low + d$"

in the above program can be replaced by the repetitive construct of the linear search

```
do X(i) < ms.low + d → i:= i + 1 od
```

.

In the time and space requirements of its execution, the resulting program is linear in N . None of the 44 students found this solution. They had three hours. Although the outcome was disappointing, the exercise served its purpose very well, and W.H.J.Feijen deserves our compliments.

8 February 1979

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow