Some Frolics.

I) Given a positive integer N, write a program to determine the number of ways in which N can be written as the sum of consecutive positive integers.

II) The syntactic category C is defined as

$$\langle C \rangle ::= \{ x\, y \langle C \rangle z \}$$

Here x, y and z are (terminal) characters, and the construct $\{ \cdots \}$ stands for zero or more successions of the enclosed.
Given an array $t(0..3N-1)$, $N \geq 0$, of characters such that $t(i) = x \;\vee\; t(i) = y \;\vee\; t(i) = z$ for all $i: 0 \leq i < 3N$, write a program that establishes whether or not the character sequence $(t(0), t(1), \ldots, t(3N-1))$ belongs to the syntactic category C.

III) Consider the set H of bit sequences that can be formed according to the rules

a) 0 belongs to H ;
b) if both h_0 and h_1 belong to H, so does the concatenation
$1 \,.\, h_0 \,.\, h_1$;
c) only those sequences that can be formed by applications of the rules a) and b) belong to H.

Given an array $h(0..2N)$ of bits, $N \geq 0$, write a program to determine whether or not the sequence $(h(0), h(1), \ldots, h(2N))$ belongs to the set H.

*

ad I) We observe that for each positive k the number of ways in which N can be written as the sum of exactly k consecutive positive integers is at most one. Is it amazing therefore that we start analysing when this is possible?
The simplest sequence of k consecutive positive integers we can think of is the sequence

$$(1, 2, \ldots, k)$$

which has, thanks to the young Gauss, as its sum

$$\tfrac{1}{2} k (k+1) .$$

The simplest sequence but one of length k definitely is

$$(1+1, 2+1, \ldots, k+1) ,$$

which therefore has a sum equal to

$$\tfrac{1}{2} k (k+1) \quad + \quad 1 . k \ .$$

And thus we observe that N can be written as the sum of k consecutive positive integers if and only if

$$N - \frac{1}{2}k(k+1)$$

is a non-negative multiple of k.

So, here is the program

```
k, e:= 1, N-1  {e = N - ½k(k+1)} ;
c:= 0;
do  e ≥ 0 →
        if e mod k = 0 → c:= c+1
        ▯ e mod k > 0 → skip
        fi;
        k, e:= k+1, e-(k+1)
od;
print (c)
```

*

ad II) We observe that a necessary condition for string t to belong to the category C is that

a) each occurrence of x in t is followed by an occurrence of y in t, i.e.

$$(\underline{A}\, l: 0 < l < 3N: \ t(l-1) \neq x \ \vee \ t(l) = y) \ \wedge \ t(3N-1) \neq x,$$

b) each occurrence of y in t is preceded by an occurrence of x in t, i.e.

$$t(0) \neq y \ \wedge \ (\underline{A}\, l: 0 < l < 3N: \ t(l) \neq y \ \vee \ t(l-1) = x).$$

Let us check first whether or not sequence t satisfies this condition, i.e. --Combining a) and b)-- we wish to establish

$$
\begin{aligned}
R:\quad cc = \;(\;& t(0) \neq y \\
\wedge\;& (\underline{A}\, l: 0 < l < 3N: (t(l-1) = x) = (t(l) = y)) \\
\wedge\;& t(3N-1) \neq x \\
)&,
\end{aligned}
$$

which can be done by what could be called "ein Stammprogramm" (a program establishing the usual postcondition in which the usual constant is replaced by the usual variable so as to obtain the usual invariant.)

```
i:= 1; cc:= (t(0) ≠ y);
do i ≠ 3*N ∧ cc →
        cc := ((t(i-1) = x) = (t(i) = y));
        i:= i+1
od;
cc := (cc ∧ t(3*N-1) ≠ x)
```

The disadvantage of this program is that it doesn't work for $N \leq 0$, but in the meantime we have learned how to manage that: we introduce an additional variable, "presym" say, such that presym = $t(i-1)$ and we include the condition $t(0) \neq y$ in the universal quantification by formally defining $t(-1) \neq x$, hence $t(-1) = y$.

The massaged program thus becomes

```
i, presym := 0, y; cc := true;
do i ≠ 3*N ∧ cc →
      sym := t(i);
      cc := ((presym = x) = (sym = y));
      i, presym := i+1, sym
od;
cc := (cc ∧ presym ≠ x)
```

+

Once the truth of cc has been established the characters x and y are indissolubly paired and all of a sudden the category C collapses into "ein Klammergebirge" with the pair xy as opening bracket and z as closing bracket, and without further justification of details we postfix the above program with

```
i, h := 0, 0;
do i ≠ 3*N ∧ h ≥ 0 ∧ cc →
      sym := t(i);
      if sym = x → h := h+1
      ▯ sym = y → h := h+1
      ▯ sym = z → h := h-2
      fi;
      i := i+1
od;
c := (cc ∧ h = 0)
```

And here we see another advantage of the previous massaging process because now the two subsequent repetitive clauses can be gracefully combined into

```
i, presym, h := 0, y, 0;   cc := true;
do  i ≠ 3×N  ∧  cc  ∧  h≥0 →
        sym := t(i);
        if sym = x  →  h := h+1
        [] sym = y  →  h := h+1
        [] sym = z  →  h := h-2
        fi;
        cc := ((presym = x) = (sym = y));
        i, presym := i+1, sym
od;
c := (cc ∧ h=0 ∧ presym ≠ x)
```

Acknowledgement: J.L.A. van de Snepscheut is acknowledged for his suggestion to combine the requirements a) and b) immediately, instead of dealing with them separately.
(End of Acknowledgement.)

*

ad III) We have to establish

R0: $hh = ((h(0), h(1), \ldots, h(2N))$ belongs to $H)$,

so let us examine H, the set of H-sequences.

Rule c) for the formation of H-sequences tells us that the rules a) and b) tell us everything about H-sequences, so that we can forget about rule c).

In particular a) and b) tell us that H-sequences are not empty, so that they contain a first element.
If an H-sequence begins with 0 its tail is empty (rule a)) and if it begins with 1 its tail is a concatenation of two H-sequences (rule b)).
So, in view of R0, mathematical induction urges us to consider the relation

$$Ch(i,c): \quad ((h(i), h(i+1), \ldots, h(2N)) \text{ is a concatenation of } c \text{ H-sequences}),$$

defined for $0 \le i \le 2N+1$ and $c \ge 0$.

One advantage of Ch is that it is easily evaluated if $(i = 2N+1 \lor c = 0)$:

$$Ch(i,c) = (i = 2N+1 \land c = 0) .$$

Another advantage of Ch is that R0 can be rephrased as

$$R1: \quad hh = Ch(i,c) ,$$

provided

$$P: \quad Ch(i,c) = Ch(0,1) \land 0 \le i \le 2N+1 \land c \ge 0$$

holds.

Finally,

(0) $\quad h(i) \neq 0 \quad \vee \quad (Ch(i,c) = Ch(i+1, c-1)) \qquad$ for $c > 0$,
(1) $\quad h(i) \neq 1 \quad \vee \quad (Ch(i,c) = Ch(i+1, c+1)) \qquad$ for $c \geq 0$

holds, and here is the program:

```
i, c := 0, 1 {P};
do i ≠ 2×N+1 ∧ c ≠ 0 →
      if h(i) = 0 → c := c-1   { Ch(i+1, c) = Ch(0,1),
                                  see (0) and P }
      ▯ h(i) = 1 → c := c+1    { Ch(i+1, c) = Ch(0,1),
                                  see (1) and P }
      fi;
      i := i+1 {P}
od;
hh := (i = 2×N+1 ∧ c = 0).
```

<u>Note</u>: The "H" of H-sequence is the last initial of Huffman.
(End of Note.)

*

W.H.J. Feijen,
Eindhoven.
April 2, 1981.