Function \texttt{partial\_sum}\,(a) on \( a = [a_1, a_2, a_3, \ldots, a_n] \)
returns \( b = [a_1, a_1 + a_2, a_1 + a_2 + a_3, \ldots, a_1 + a_2 + \ldots + a_n] \)

\[
\text{mid\_1 }
\]

mode(0); warning("off")
function \texttt{b=partial\_sum}\,(\texttt{a})
    \texttt{b=a}
    \texttt{... delete this line}
endfunction

\[
\text{a=1:8}
\]
\[
\text{partial\_sum}\,(\texttt{a})
\]

\[
\text{mid\_2 }
\]
\texttt{divisor\_sum}\,(n) = \text{the sum of divisors of } \texttt{n}

\[
\text{for } \texttt{i=1:6; printf("i=%i,divisor\_sum=%i\n",i,divisor\_sum(i)) end}
\]
mid_3 hailstorm program. n=7. Infinite while (%t) loop:

Prints n.

Returns if n=1.

If n is even, sets n equal to n/2 and then continues.
If n is odd, sets n equal to 3n +1 and then continues.

//hailstorm
mode(0);warning("off")
n=7;
while(%t)
    printf("%i ",n)
end
Function my_factorial(n) = n!

for i=1:10;
    printf(" n=%i n!=%i\n",i,my_factorial(i))
end

roll() rolls a die randomly giving 1,2,3,4,5,6.

for i=1:20
    printf(" %i",roll())
end