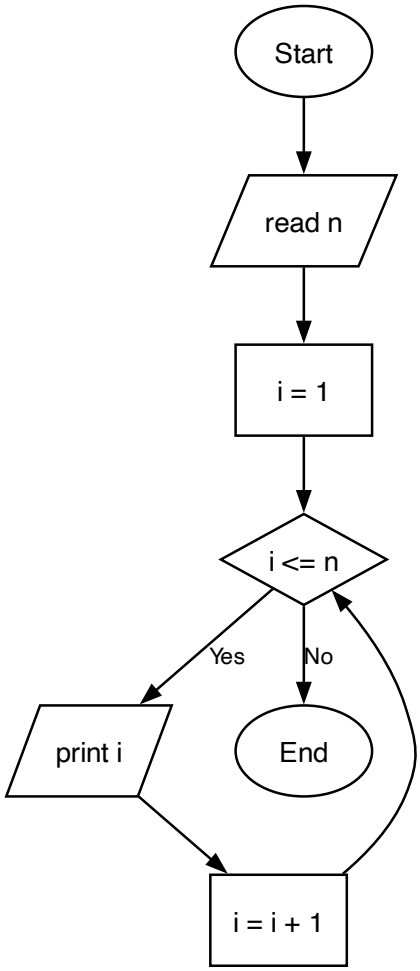
Problem 1.

Trace this algorithm for $n = 5$, and complete the trace table.



n	i	output

Problem 2.

> *Ask the user for a number n, then print all the numbers from n down to 1, one at a time (so for n = 5 it prints 5, 4, 3, 2, 1).*

Draw a flowchart for this algorithm:

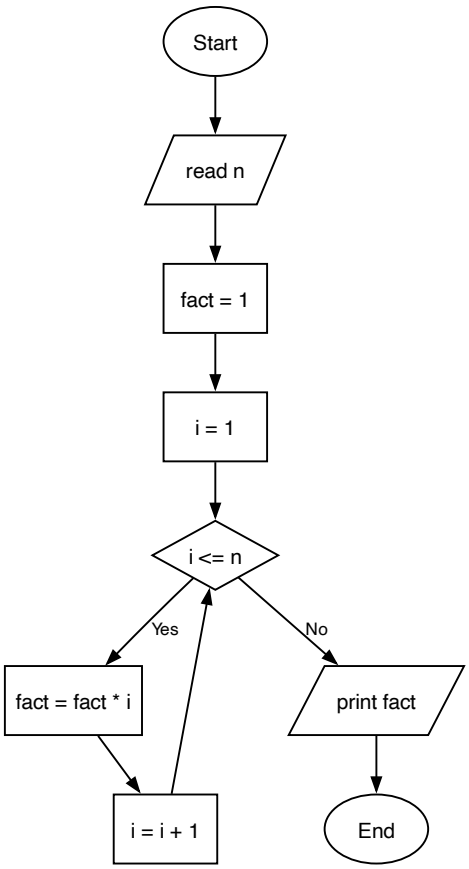
Problem 3.

> *Ask the user for a number n and print its multiplication table from n × 1 up to n × 10 — that is, print the values n×1, n×2, n×3, …, n×10.*

Draw a flowchart for this algorithm:

Problem 4.

Trace this algorithm for $n = 5$, and complete the trace table.

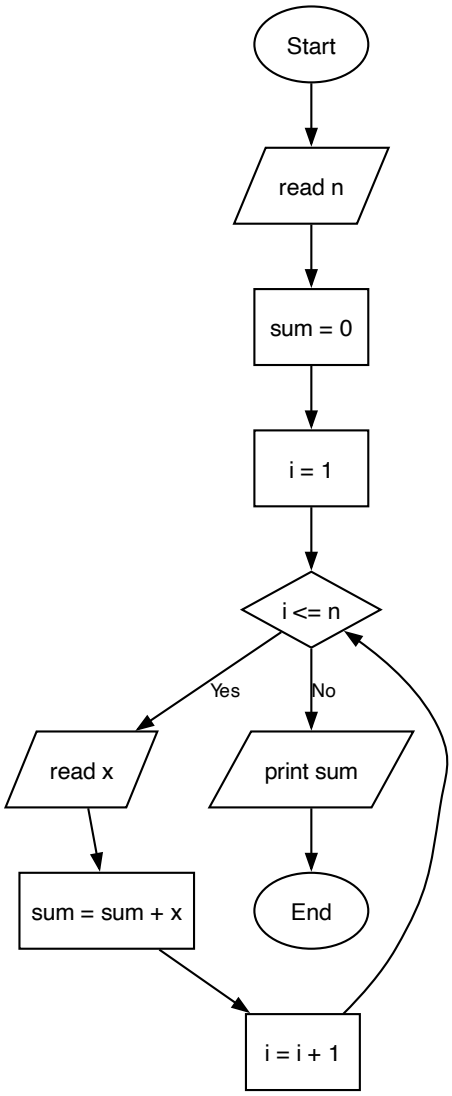


n	fact	i	output

Problem 5.

This program reads how many numbers there are (n), then reads that many numbers one by one and prints their total.

Trace this algorithm for $n = 3$, where the values of x entered are 10, 20, 30 (in that order), and complete the trace table.



n	sum	i	x	output

Problem 6.

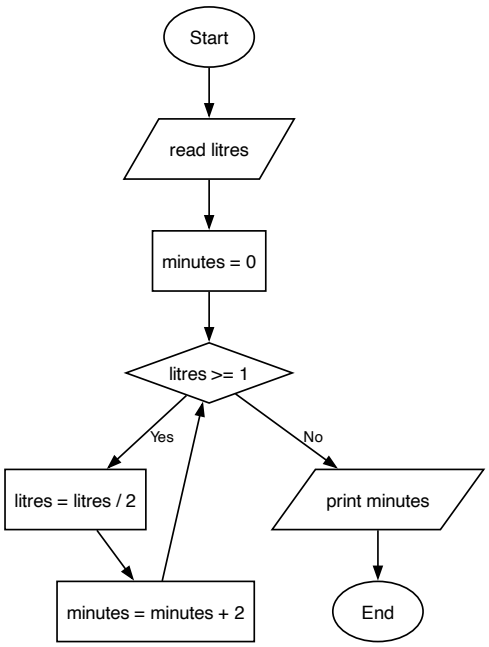
Ask the user for a count n, then read n numbers one by one and print the largest of them. (Hint: remember the first number as the "largest so far", then compare every later number against it and update it when you find a bigger one.)

Draw a flowchart for this algorithm:

Problem 7.

A water tank holds some litres of water. Every 2 minutes, exactly half of the water currently in the tank drains out. The program keeps going until less than 1 litre is left, and prints how many minutes that took. (Notice that the tank never becomes completely empty — can you see why?)

Trace this algorithm for $litres = 64$, and complete the trace table.



litres	minutes	output

Problem 8.

An investment of P rupees grows by 10% every year — so each year the amount becomes 1.1 times what it was the year before. Print how many years it takes for the investment to at least double (that is, to reach 2 × P).

Draw a flowchart for this algorithm: