1. **Problem descriptions**. Often a problem will be described in vague terms, and you need to turn it into an unambiguous specification for what the result should look like for all possible inputs.

   Write down a description for each of the following problems. State what inputs are needed, what outputs should be produced, and the main calculations needed to solve this problem. See the examples in the lecture notes. **Note**: You don't need to find an answer, just list the things you'd need to know and work out.

   (a) You're buying a pizza and want to maximize the value for money. Given a takeaway menu, find the pizza with the best area to price ratio

   (b) Given the dimensions of an Egyptian pyramid and density of stone, how many 10-tonne trucks would be needed to move it in one trip?

   (c) You plan to breed hippos. How many should you buy now in order to retire with a sufficient number (100 say)?

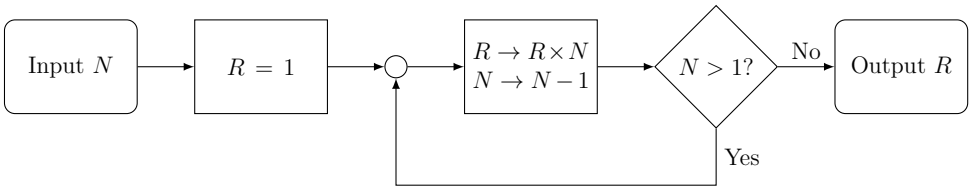
2. **Flow charts**. Using the symbols in the lecture, draw flow charts for solving the following problems:

   (a) Read two numbers from the user, then print them out in decreasing order

   (b) Find the sum of first 50 natural numbers $(1, 2, \ldots, 50)$

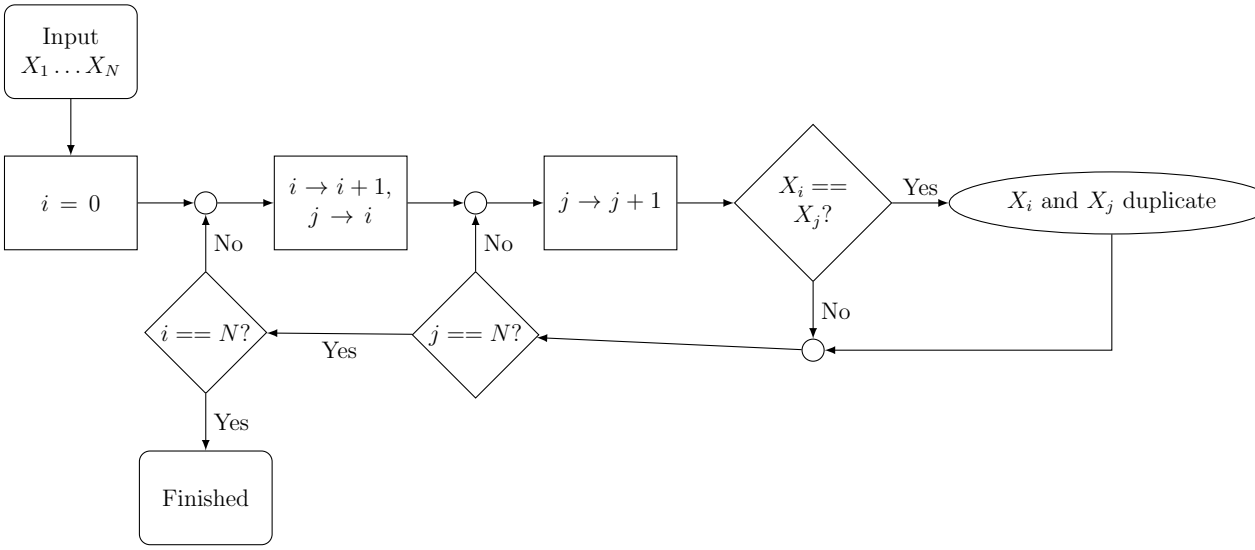
3. **Algorithm performance**. For each of the algorithms below, state how the time taken to run them will depend on the size of the input N. For example, if the amount of work to do is proportional to the square of the problem size, then an algorithm is $O(N^2)$.

   (a) Calculate the factorial $N!$ of a number N



P.T.O.

(b) Find duplicates in a list of N numbers



(c) Guess a number m between 1 and N. Each time the program makes a guess g, it can test whether it is larger, smaller or equal to m.