Algorithms.

An algorithm is defined as a finite set of rules, which gives a sequence of operations for solving a specific type of problem.
In other words, algorithm is a step-by-step procedure for solving problems.

An algorithm has following five important features:

(1) Finiteness

(2) Definiteness

(3) Completeness

(4) Input

(5) Output

(1) **Finiteness:** An algorithm should always terminate after a finite number of steps.

(2) **Definiteness:** Each step of algorithm should be precisely defined. This means that the rules should be consistent and unambiguous.

(3)**Completeness:** The rules must be complete so that the algorithm can solve all problems of a particular type for which the algorithm is designed.

(4)**Input:** An algorithm has certain inputs.

(5)**Output:** An algorithm has certain outputs which are in specific relation to the inputs.
An important consideration for an algorithm concerns its efficiency. Some algorithms are far more efficient than others in that, when programmed, one may require fewer steps or perhaps less memory than another and will therefore, be more satisfactory or economical in actually solving problems on a computer. We shall often deal with considerations of this type in the subsequent work.
In the development of an algorithm, sequence, selection and repetition (or interaction) play an important role.

(1)**Sequence:** Suppose that we want to find the value of the expression $a^3 + 4ab + b^2$ for given values of a and b. Algorithm (i.e., step by step procedure) for achieving this will consist of steps given in fig. to be carried out one after the other.

1. Get the value of *a*
2. Get the value of *b*
3. Calculate a^3, call it *S*
4. Calculate $4ab$, call it *T*
5. Calculate b^2, call it *V*
6. Find the sum $S + T + V$, call it *M*
7. Write the value of *M* as answer.

Steps of an algorithm to evaluate $a^3 + 4ab + b^2$, given the values of *a, b*

This algorithm, you will agree, is very straightforward, consisting of simple steps which are to be carried out one after the other. We say that such an algorithm is a sequence of steps, meaning that

(i) At a time only one step of the algorithm is to be carried out.

(ii) Every step of the algorithm is to be carried out once and only once; none is repeated and none is omitted.

(iii) The order of carrying out the steps of the algorithm is the same as that in which they are written.

(iv) Termination of the last step of the algorithm indicates the end of the algorithm.

Here afterwards we shall follow the convention that (i) the successive steps in a sequence will be written on successive lines and hence (ii) steps will not be necessarily numbered as they are in fig.

(2) **Selection:** An algorithm which consists of only a sequence, is not sufficient for solving any type of problem. Let us consider the problem of solving an equation of the type $m + nx = r$ (where m, n, r are given integers) for integral values of x. We immediately use laws of algebra to find $x = (r - m) \div n, n \neq 0$. Let us call an algorithm that works for only some (not necessarily all) possible sets of input values, a semi-algorithm.

Semi-algorithm (for the above problem) :

Step 1: Get the values of m, r and n.

Step 2: Subtract m from r, call this difference b.

Step 3: Divide b by n; print this result as the value of x.

The above steps are certainly efficient, As an example, let $m = 9, n = 5$ and $r = 24$, in which case we have $9 + 5x = 24$. Then in step 2, we have $24 - 9 = b$ i.e., $b = 15$ and in step 3, we have $\frac{b}{n} = \frac{15}{5} = 3$, and so we print x = 3.

The above steps have two fatal flaws, however. First, if n equals 0, then either $m = r$ and x can have any integral value, or $m \neq r$ and no solution is possible i.e., there is no integer x which may satisfy the given equation. Second, if there is a non-zero remainder when b is divided by n then again there is no integer x which may satisfy the given equation. So we must modify our algorithm to deal with all such situations as may arise. Given below is the modified algorithm which suits all the possible situations that may arise.

Step 1 : Get the value of *m*, *n* and *r*

Step 2 :**If** $n = 0$ and $m = r$

then go to step 7

else go to step 3

Step 3 :**If** $n = 0$ and $m \neq r$

then go to step 6

elsego to step 4

Step 4 :**Subtract***m* from *r*, call this difference b (*i.e.,* $b = r - m$)

Step 5: **Divide***b* by *n*;

If there is a remainder

thengo to step 6

else print the value of $\frac{b}{n}$, which is the required value of *x*.

Step 6: **Print** 'No integer satisfies this equation'. Stop

Step 7 :**Print** 'Any integer satisfies this equation'. Stop

Algorithm to solve an equation of the form
m + *nx* = *r*, where *m, n, r, x* are integers

The above algorithm provides the person or computer that will execute the algorithm with an ability to choose the step to be carried out depending upon the values of m, n and r (and subsequently, the value of b). This ability is called selection. The power of selection is that it permits that different paths could be followed, depending upon the requirement of the problem, by the one who executes the algorithm.

In the above algorithm, selection is expressed by using the special words 'if, 'then', 'else'. Further, it may be noted that all that is written using these special words (once) constitutes one step. Note the way it is

written. Nothing appears below the word 'if' till that step is over. This is known as indentation. The words 'then' and 'else' come with exactly same indentation with respect to the word 'if'.

(3) **Iteration or Repetition:** In forming an algorithm certain steps are required to be repeated before algorithm terminates after giving an answer. This is known as iteration or repetition.

Let us consider the problem of finding the just prime number greater than a given positive integer. The following list of steps shows the step by step procedure to be followed for solving the problem.

Consider the given integer

S { add 1 to it
test new number for primeness
if it is prime, }
then write it down and stop
S { **else** add 1 to it
test new number for primeness
if it is prime, }
then write it down and stop
S { **else** add 1 to it
test new number for primeness
if

Algorithm for finding a prime number
greater then a given positive integer

We see in the above procedure that the steps

"add 1 to it

test new number for primeness

if it is prime

then write it down and stop ”

are repeated again and again till (after a finite number of repetitions) we get a prime number and print it. If this sequence (which involves a decision also) is denoted by S, then S is repeated again and again till, we get the result and print the result. This is technically known as iteration or repetition. The way of writing adopted in fig. poses a problem as we do not know the number of times S is repeated. This number depends upon the given positive integer. The difficulty presented above is overcome by introducing a new way of writing iterations in algorithms. The algorithm shown in fig. is (in new ways) then written as shown below

Consider the given number

repeat

add 1 to it

Or

Consider the given number

add 1 to it

while the new number is not prime

Two different ways of writing iteration occuring in fig.