ALGORITHMS and FLOWCHARTS

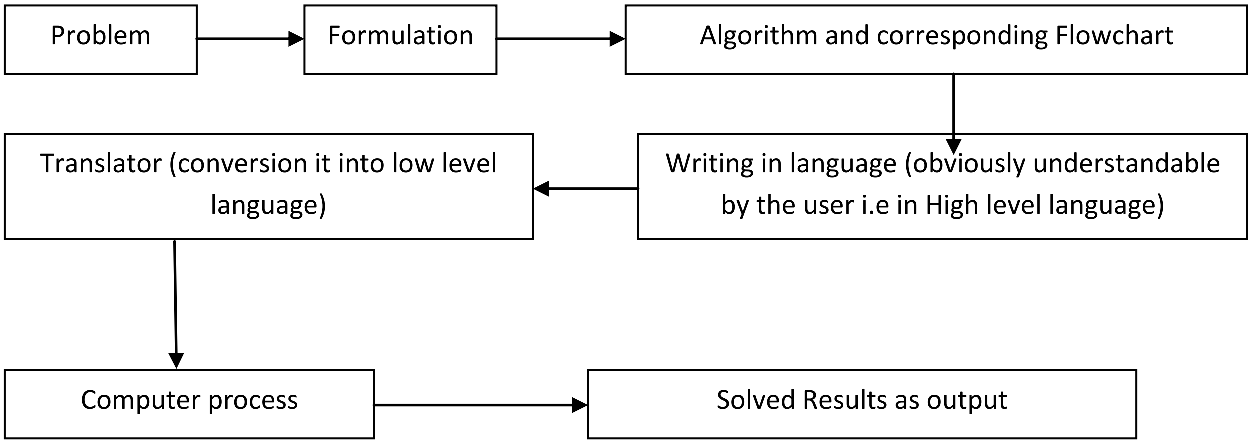
Mathematical Methods-II Lab, Programming in C language

This is very important to know the concepts of algorithm and flowchart before writing programs in any language. If you make your logic clear to solve any problem with the use of algorithm and flowchart, then it is very easier to write in any computer language. So in this module I am just trying to make your concept clear. The Video of this class is available in our college website under E-content.

Dr. Saumen Charaborty

To solve any problem through computer we have to take help of computer program. Computer programs are collections of instructions that tell a computer how to interact with the user, interact with the computer hardware and process data and finally solve the said problem.

The total process can be summaries as follows,


Problem
Formulation
Algorithm and corresponding Flowchart
Translator (conversion it into low level language)
Writing in language (obviously understandable by the user i.e in High level language)
Computer process
Solved Results as output

Algorithm:

The word 'Algorithm' has originated from the word 'algorism', which refers to the 'art of computing'. An *algorithm*, is defined as a "well-ordered collection of unambiguous and effectively computable operations, that when executed, produces a result and halts in a finite amount of time." In other word, an **algorithm** is a step-by-step procedure to solve a given problem.

Characteristics of an Algorithm

- Well-ordered: the steps are in a clear order
- Unambiguous: the operations described are understood by a computing agent without further simplification
- Effectively computable: the computing agent can actually carry out the operation
- Finiteness: must terminate after a finite number of steps
- Completeness: must be general so that it can solve any problem of a particular type for which it is designed

Efficiency of an algorithm

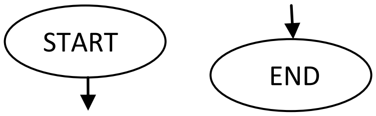
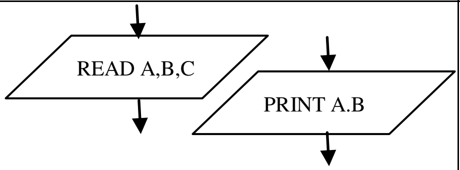
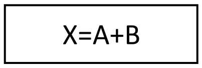
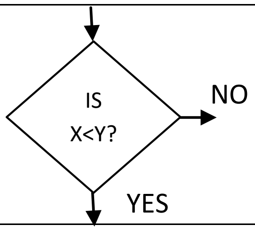
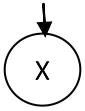
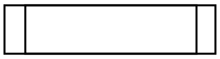
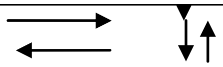
- How fast the algorithm solves the problem.
- How accurate result the algorithm can compute.
- Small error in the input data should not produce large errors in the output data.

Flowcharts:

Flowchart is a graphical tool that *diagrammatically* depicts the steps and structure of an algorithm or program. A flowchart is a diagram made up of *boxes*, *diamonds* and other shapes, *connected by arrows* - each shape represents a step in the process, and the arrows show the order in which they occur. Flowcharting combines symbols and flow lines, to show figuratively the operation of an algorithm.

Flowcharting Symbols

There are 6 basic symbols commonly used in flowcharting of assembly language programs: Terminal, Process, input/output, Decision, Connector and Predefined Process. This is not a complete list of all the possible flowcharting symbols, it is the ones used most often in the structure of Assembly language programming.

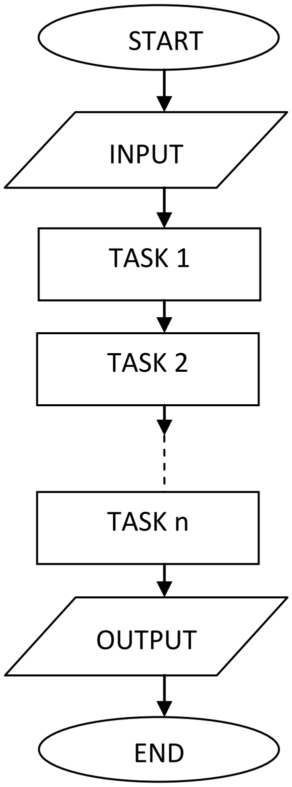
Symbol	**Name**	**Function**
	Terminal	Indicates the starting or ending of the program, process, or interrupt program.
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Process	Indicates any type of internal operation inside the Processor or Memory
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an interrupt program.
	Flow Lines	Shows direction of flow.

The Structure of a Flowchart

A flowchart of a program can be drawn by making any kind of connections but this may lead to difficulties in creating and maintaining large programs. From practical point of view, a flowchart should be drawn in an organized manner so that it can be easily read, understood and tested, thereby increasing the reliability of the program and user productivity. The method of writing a program or flowchart in this manner is known as structured programming. In 1966, computer scientists Corrado Böhm and Giuseppe Jacopini demonstrated that all programs could be written using three control structures: Sequence, Selection, and Repetition.

(i) Sequential Structure:

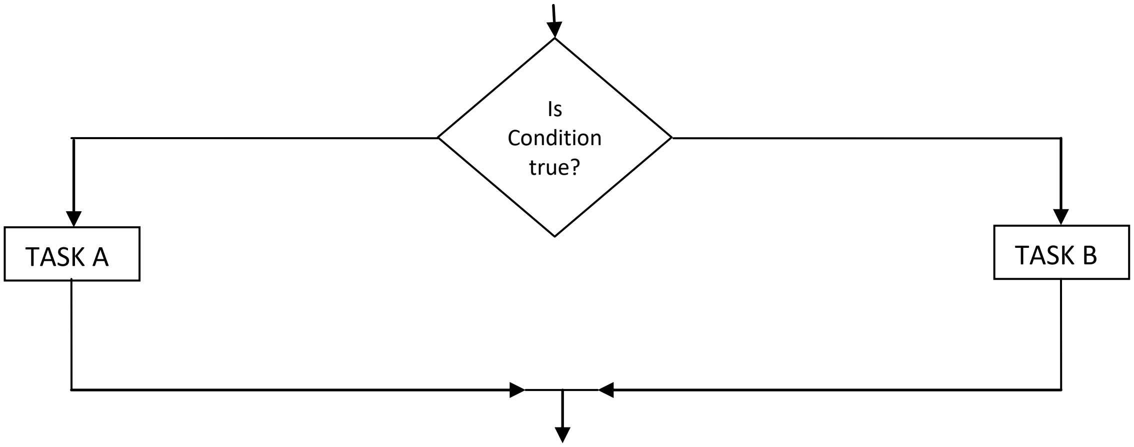
In the *sequential* structure the control flows from one step to next step. Here, one statement is executed after another and it will stop after the completion of all the steps. The flowchart is as follows;



Sequential Structure

(ii) Conditional Structure:

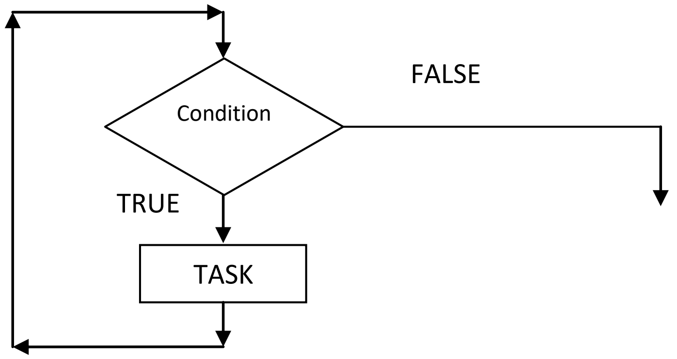
The *selection* structure is the construct where statements can executed or skipped depending on whether a condition evaluates to TRUE or FALSE. If the condition is true then one path is taken otherwise it takes the other path. This is shown as follows;



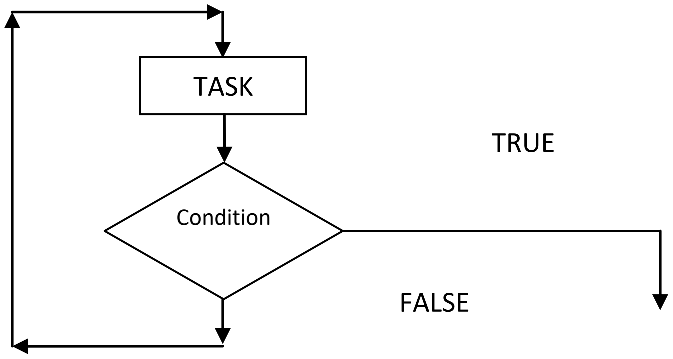
Conditional Structure

(iii) LOOP Structure:

This is also known as repetitive structure. Any repetitive structure mainly consists of a loop or repetition. Actually, in this structure one or few instruction statements can be executed repeatedly until a condition evaluates to TRUE or FALSE. It is of two types. One is DO WHILE and another is DO UNTIL.



DO WHILE

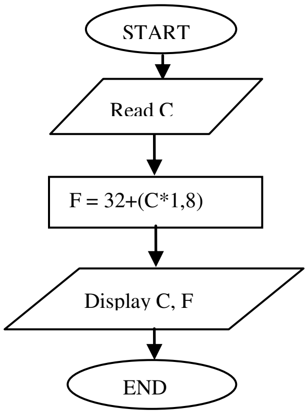


DO UNTIL

Example 1: Convert temperature from centigrade to Fahrenheit. (Hints:To convert Centigrade to Fahrenheit, multiply by 1.8 and add 32 degrees.)

```
Algorithm:  Step 1 : start
            Setp 2 : read temperature in centrigrade
            Step 3 : caluculate Fahrenheit = 32 + (centigrade * (1.8));
            Step 4 : display centigrade and Fahrenheit
            Step 5 : stop
```

Flow chart:

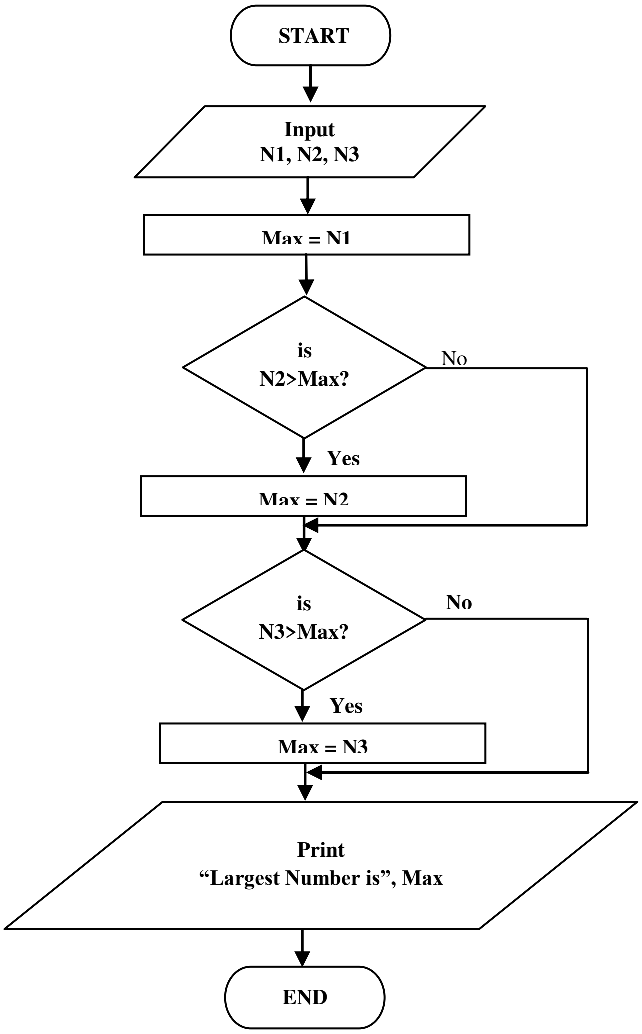


Example 2: Write down an algorithm and draw a flowchart to find and print the largest of three numbers.

Algorithm:

```
Step 1:  Input N1,N2,N3
Step 2:  Max = N1
Step 3:  If N2>Max
         then Max = N2
         endif
Step 4:  If N3>Max
         then Max = N3
         endif
Step 5:  Print  "The latgest number is:",Max
```

Flowchart:



Example 3: Write an algorithm and draw a flowchart to calculate the factorial of a number (N). Verify your result by a trace table by assuming N = 5. (Hint: The factorial of N is the product of numbers from 1 to N)

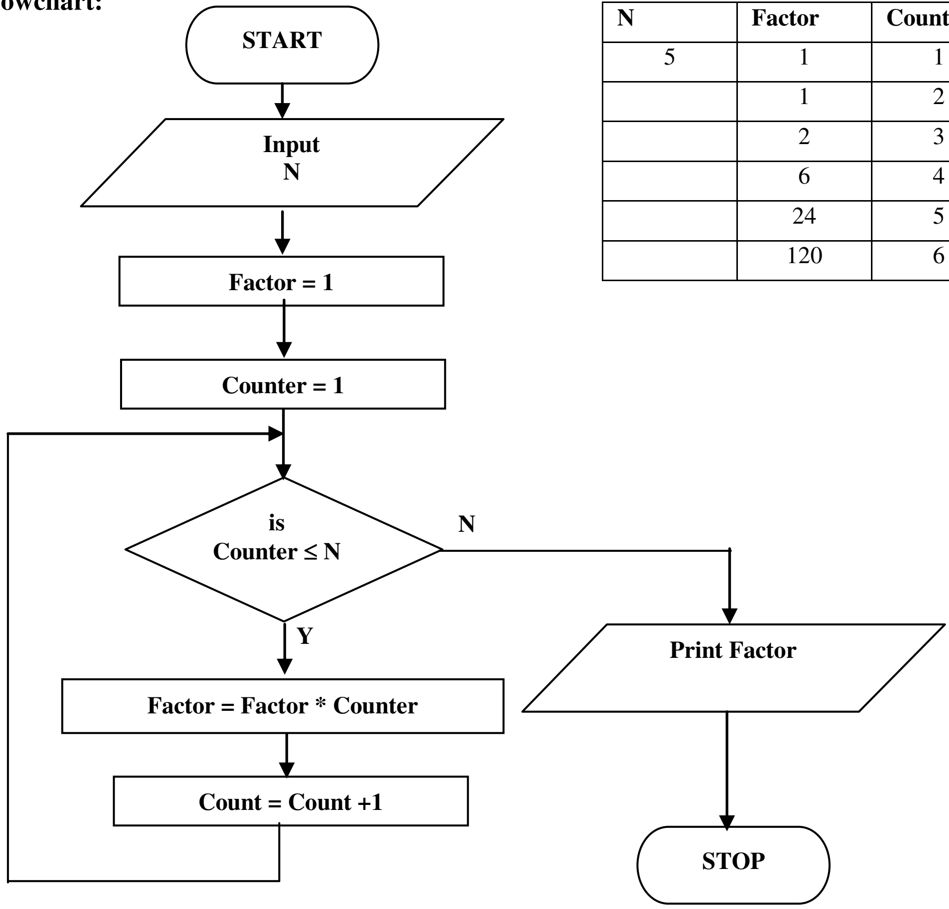
Algorithm:

```
Step 1:  Input N
         Step 2:  Factor = 1
         Step 3:  Counter  = 1
         Step 4:  While(Counter ≤ N)
                                      Repeat steps 4 through 6
         Step 5:            Factor = Factor * Counter
         Step 6:            Counter = Counter + 1
         Step 7:   Print (N, Factor)
```

Trace Table:

N	Factor	Counter	Decision	Print
5	1	1	Y	
	1	2	Y	
	2	3	Y	
	6	4	Y	
	24	5	Y	
	120	6	N	120

Flowchart:



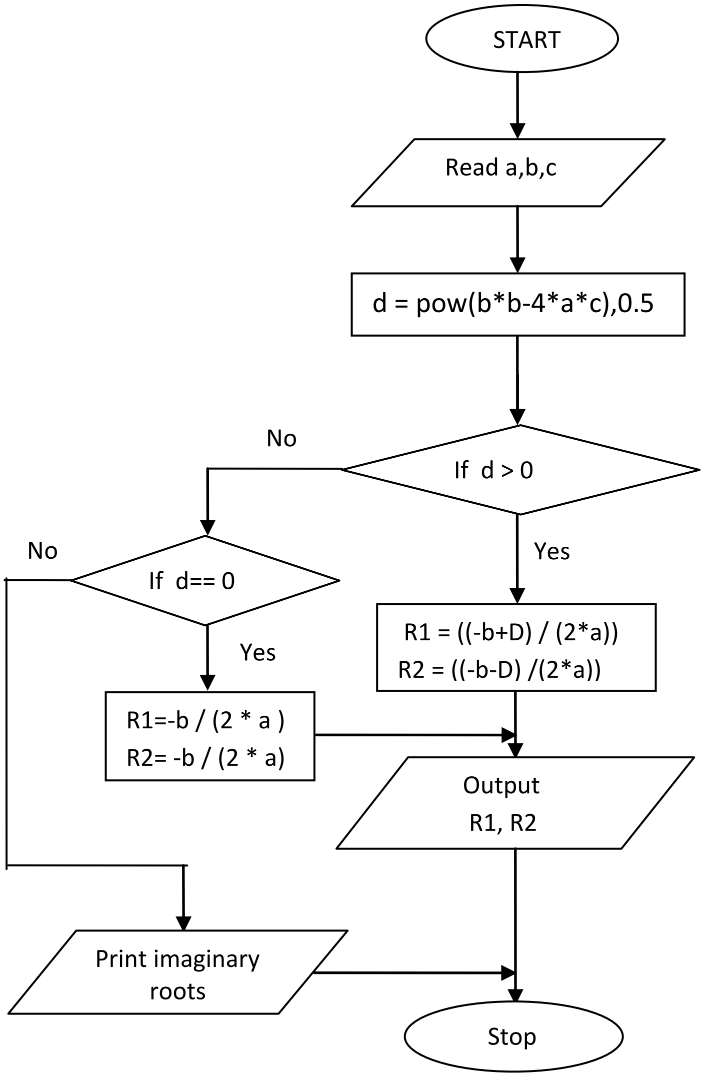
Extra Examples:

Example E.1: To find the roots of the quadratic equation. (Hints: Nature of roots of quadratic equation can be known from the quadrant $\triangle = b^2$-4ac, If b^2-4ac >0 then roots are real and unequal, If b^2-4ac =0 then roots are real and equal, If b^2-4ac <0 then roots are imaginary)

Algorithm:

```
Step 1: start
Step 2: read the a,b,c value
Step 3: if (b*b-4ac)>0 then
            Root 1= (-b+ pow((b*b-4*a*c),0.5))/2*a
            Root 2= (-b-pow((b*b-4*a*c),0.5))/2*a
Step 4: if (b*b-4ac)=0 then
            Root1 = Root2 = -b/(2*a)
Step 5: Otherwise Print Imaginary roots. Goto step 7.
Step 6: print roots
Step 7: stop
```

Flowchart:

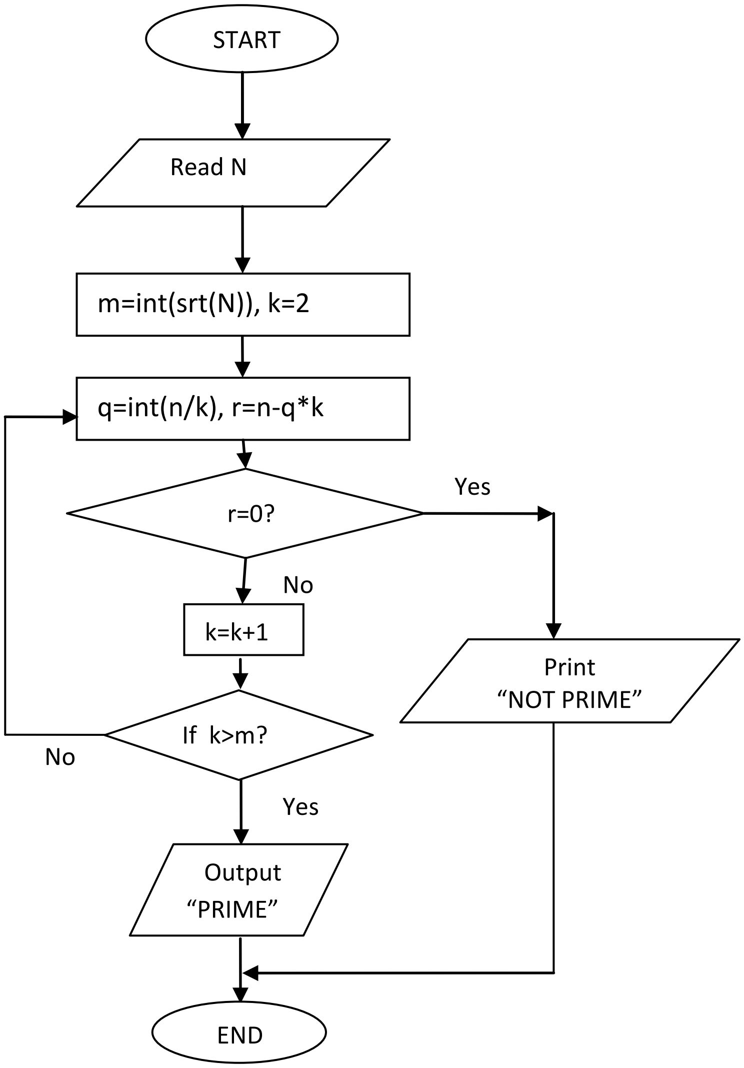


Example E.2: Write a algorithm and flowchart to check a given number is prime or not. (Hints: Prime number is a number which is exactly divisible by one and itself only, Ex: 2, 3,5,7,………;)

Algorithm:

```
Step 1: Start
Step 2: Read n
Step 3: m=integer($\sqrt{n}$), k=2
Step 4: Dividing n by k
            Find the remainder r.
Step 5: If r=0:output is not primed, goto Step 10
Step 6: k=k+1
Step 7: k>m: Exit from loop, goto Step 9
Step 8: Repeat Steps 4 to7
Step 9: Output: n is prime
Step 10: End
```

Flow chart:



Example E.3: To print prime numbers from 1 to N. (Modification of above problem)

Algorithm:

```
Step 1: Start
Step 2: Read N
Step4: i=1
Step5: While (i<=N)
        Repeat Step 6 to Step 13
Step 6: m=integer($\sqrt{i}$), k=2
Step 7: Dividing N by k
        Find the remainder r.
Step 8: If r=0 (means not primed), goto Step 13
Step 9: k=k+1
Step 10: if k>m: Exit from loop, goto Step 12
Step 11: Repeat Steps 4 to7
Step 12: Print i
Step 13: i=i+1
Step 14: End
```

(You can also modify the above problem where you print the not primed numbers instead of prime numbers)

Problem 1: Write an algorithm and draw a flowchart to calculate $\sum_{x=1}^{10} x^2$.

Problem 2: Write an algorithm and draw a flowchart to calculate the area of a triangle having the sides a, b, c repectively. (Hints: At first check triangle is possible or not)

Problem 3: Write an algorithm and draw a flowchart to multiply two integer without using multiplication operator.