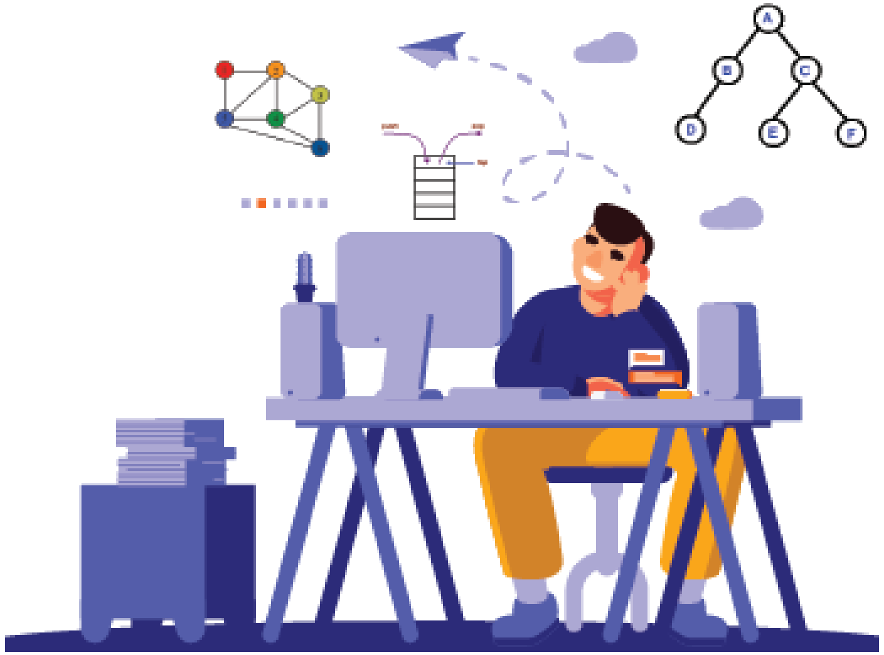
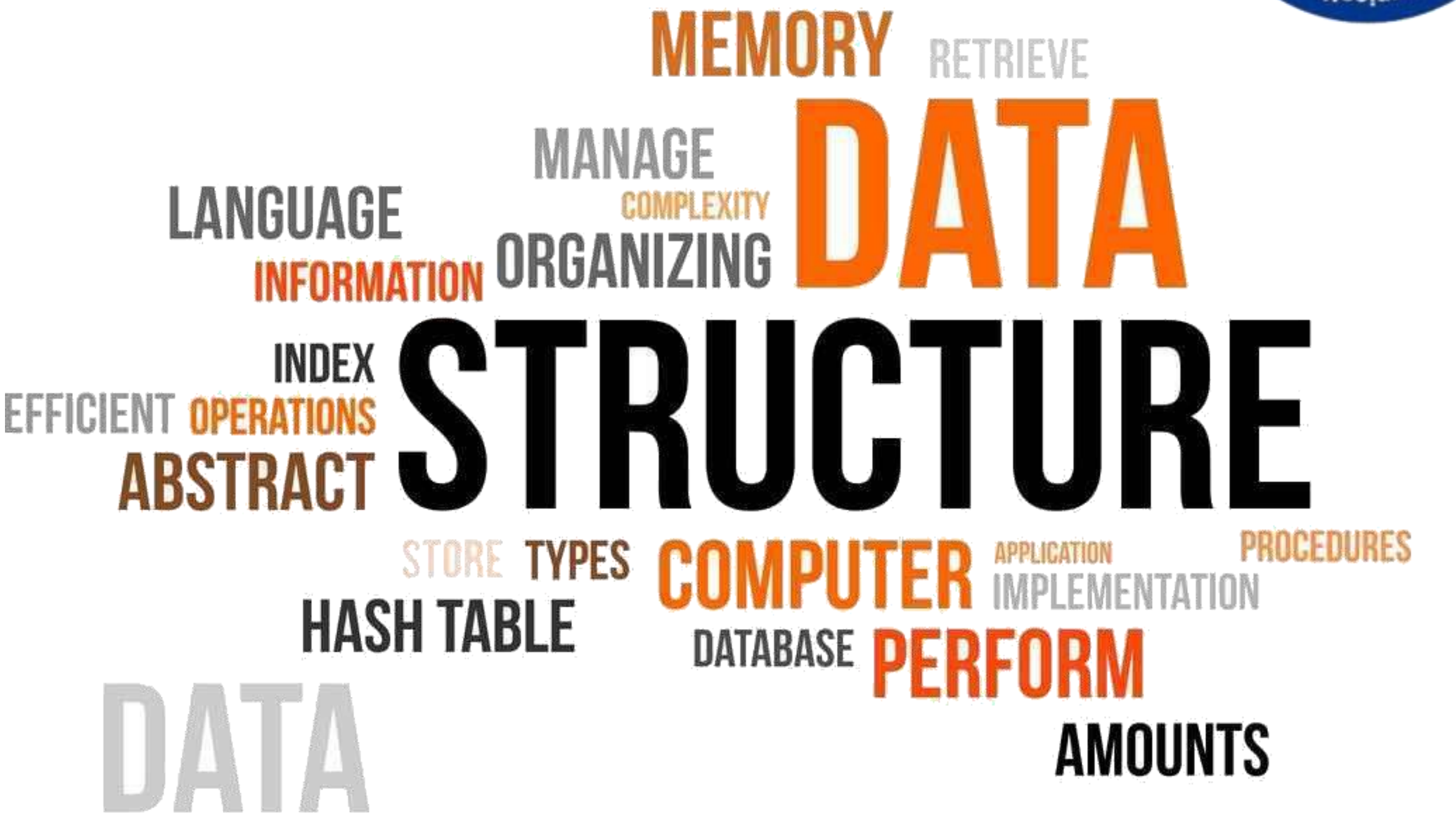


Data Structures

Course code: IT623

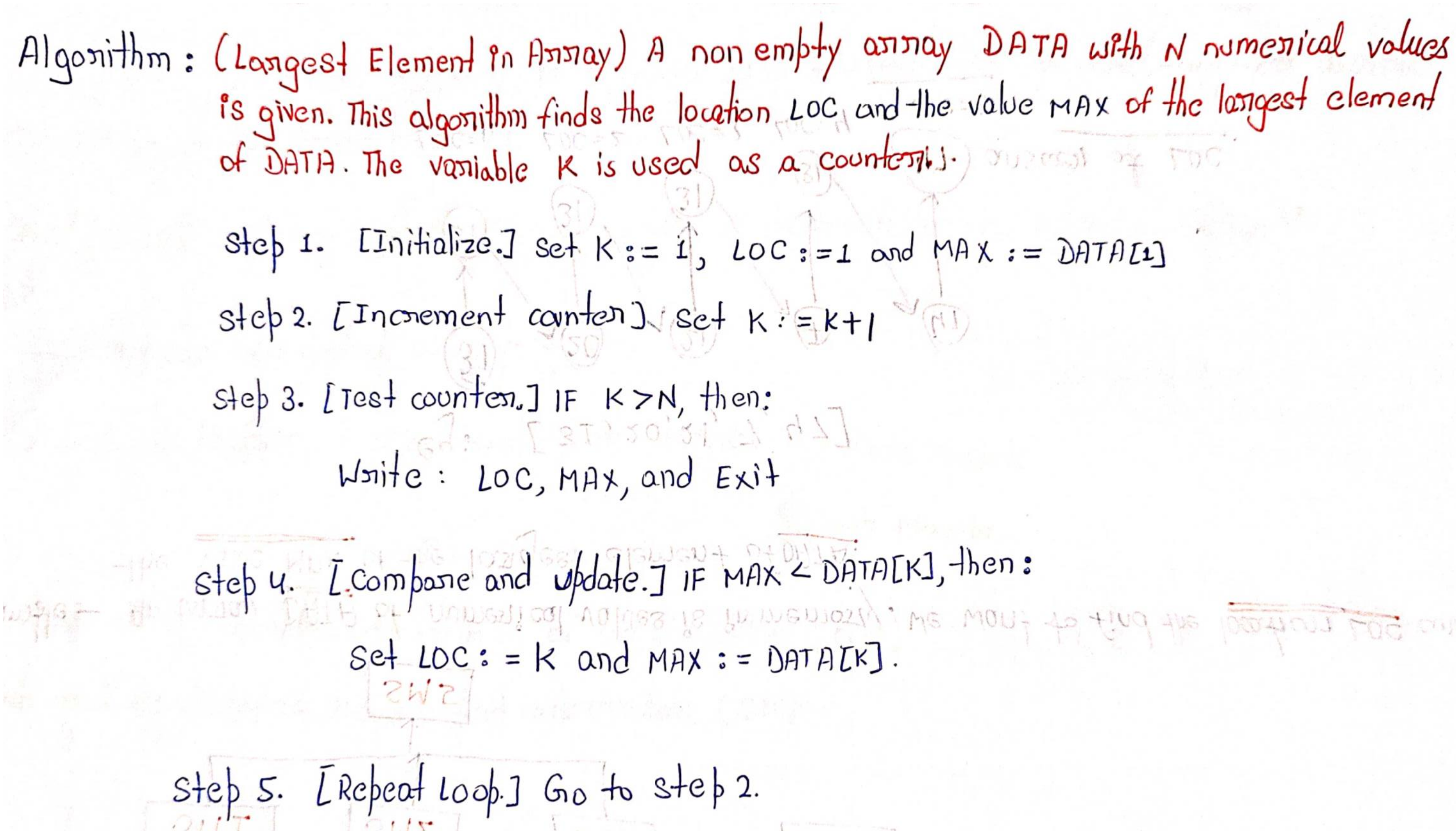


Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Disclaimer: Images are sourced from the Internet and may be copyright protected.

Lectures 4

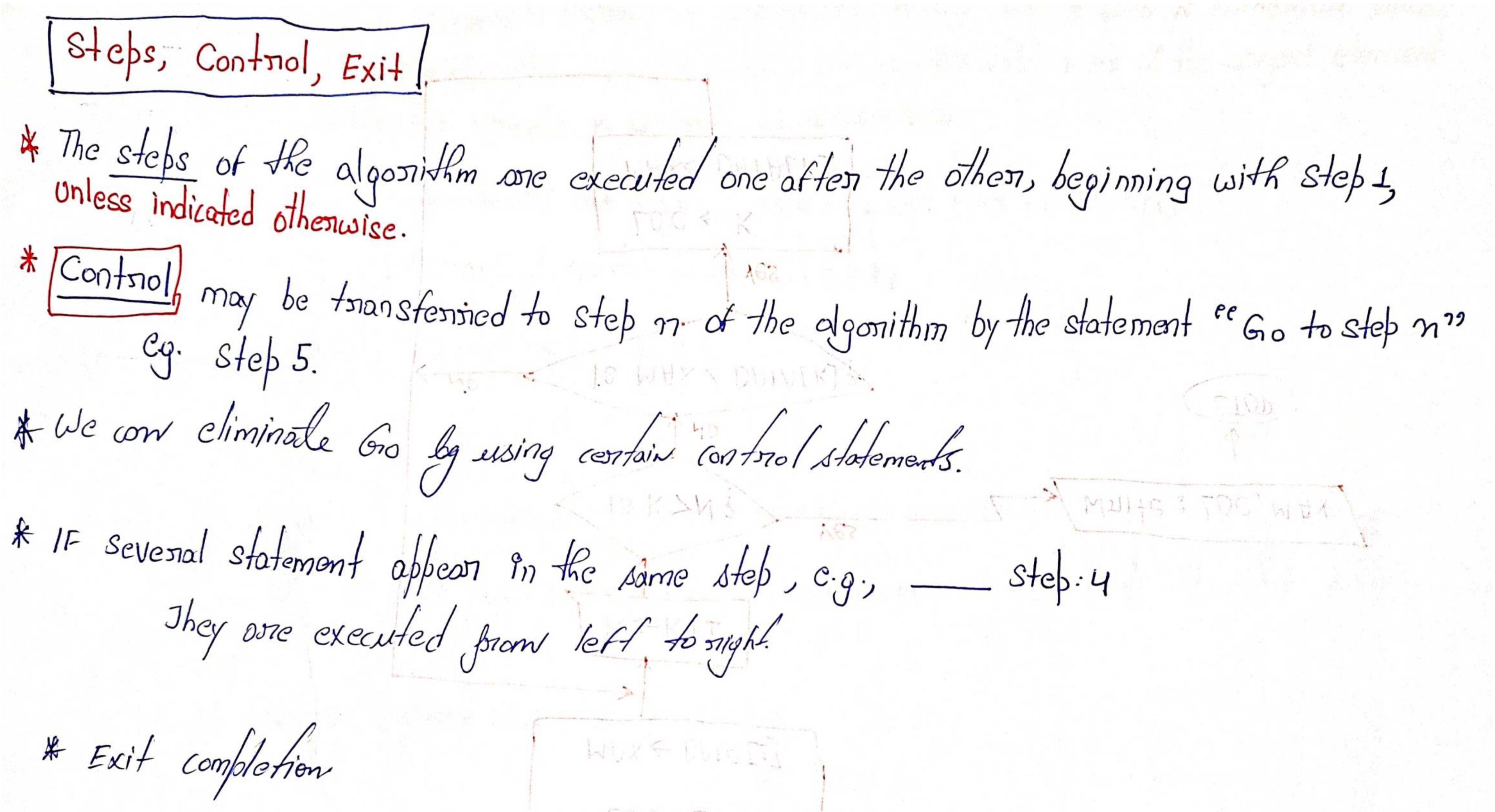
Standard Notations and common functions



Algorithm : (Largest Element in Array) A non empty array DATA with N numerical values is given. This algorithm finds the location LOC and the value MAX of the largest element of DATA. The variable K is used as a counter.

```
Step 1. [Initialize.] Set K := 1, LOC := 1 and MAX := DATA[1]

Step 2. [Increment counter] Set K := K+1

Step 3. [Test counter.] IF K > N, then:

            Write : LOC, MAX, and Exit

Step 4. [Compare and update.] IF MAX < DATA[K], then :

            Set LOC := K and MAX := DATA[K].

Step 5. [Repeat Loop.] Go to step 2.
```

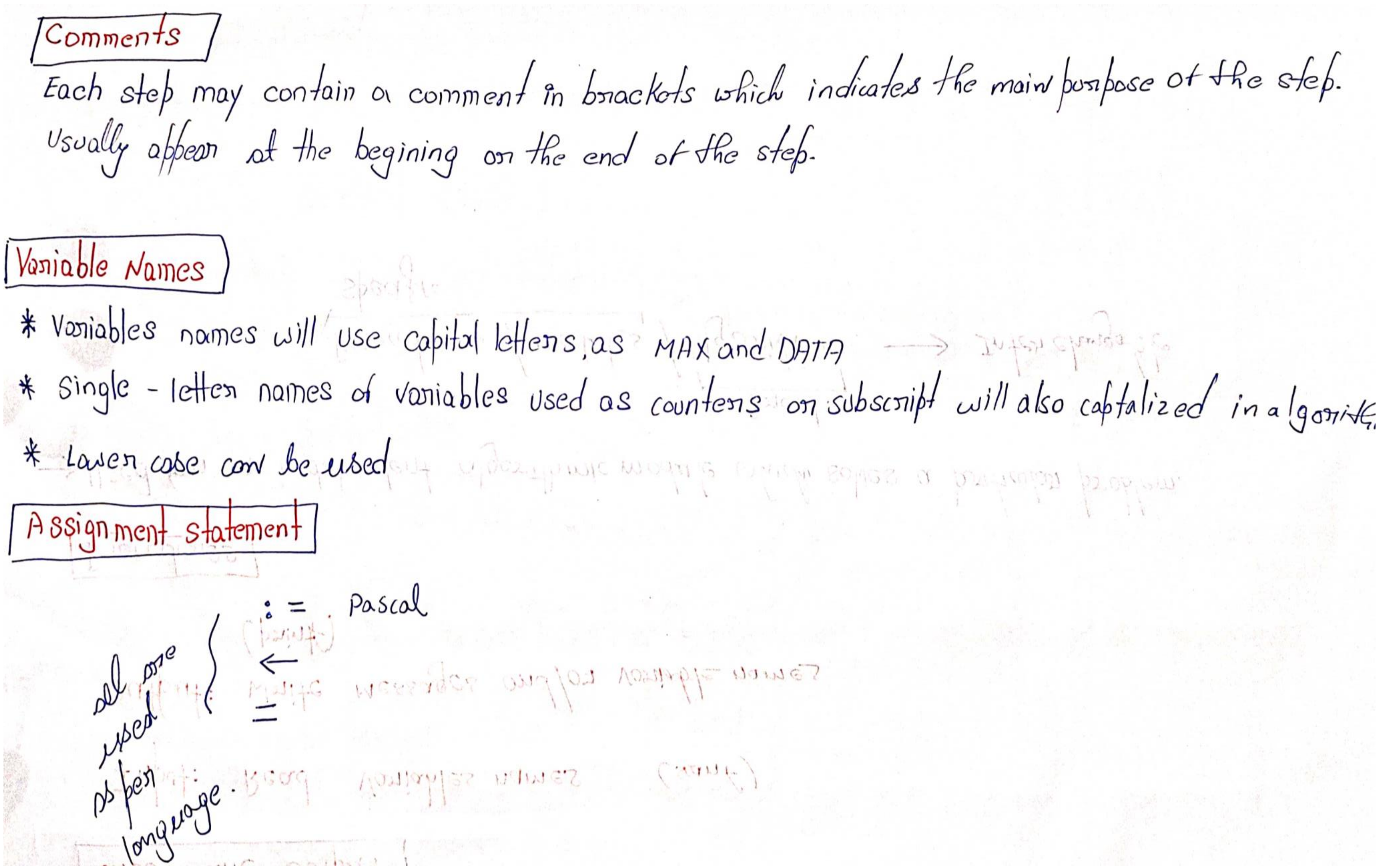
Standard Notations and common functions



Steps, Control, Exit

- * The steps of the algorithm are executed one after the other, beginning with step 1, unless indicated otherwise.
- * Control may be transferred to step n of the algorithm by the statement "Go to step n"
  e.g. Step 5.
- * We can eliminate Go by using certain control statements.
- * If several statement appear in the same step, e.g., —— step: 4
  They are executed from left to right.
- * Exit completion

Standard Notations and common functions



Comments

Each step may contain a comment in brackots which indicates the main purpose of the step. Usually appear at the begining or the end of the step.

Variable Names

* Variables names will use capital letters, as MAX and DATA

* Single - letter names of variables used as counters or subscript will also captalized in algorith

* Lower case can be used

Assignment statement

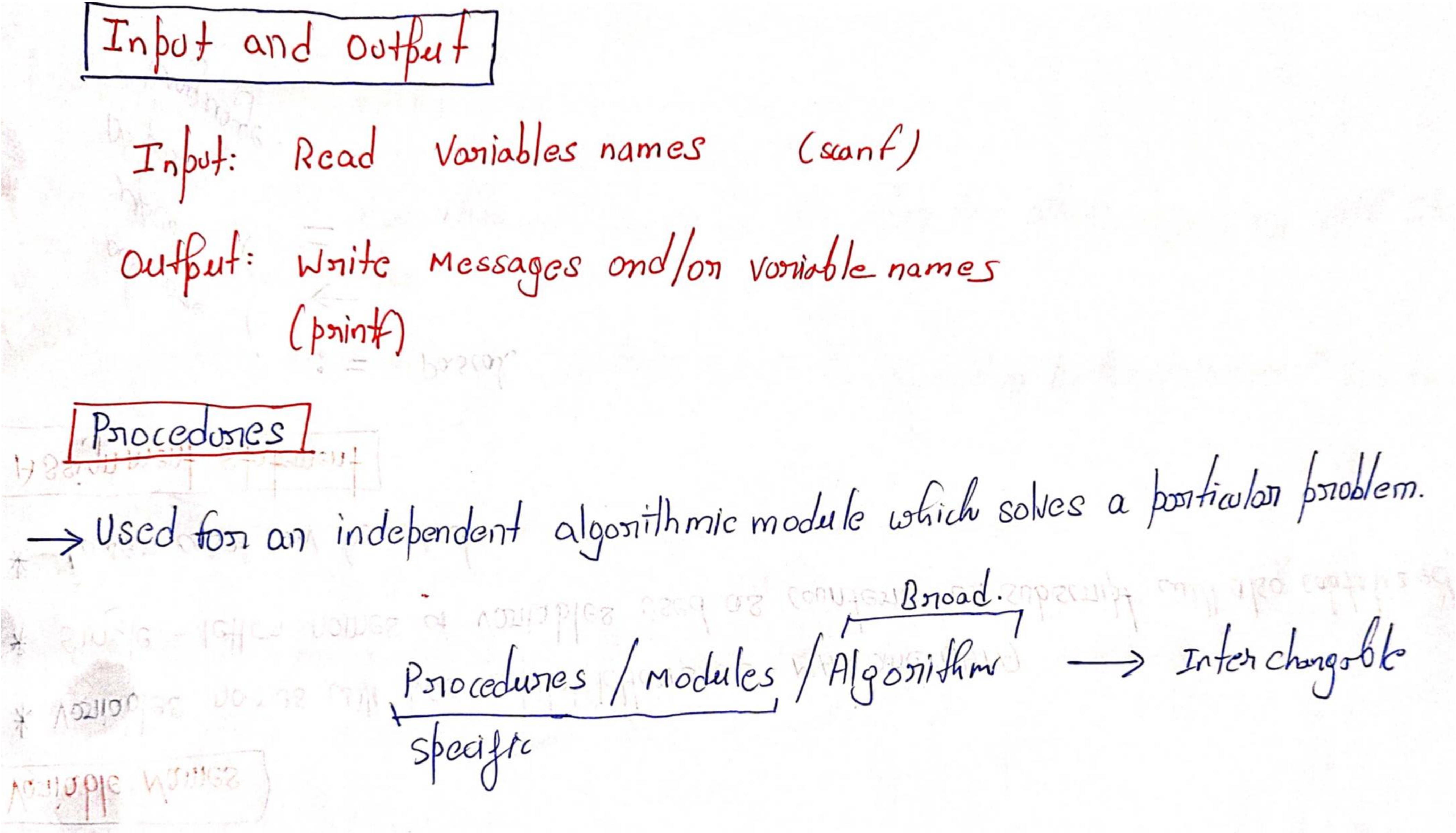
:= Pascal

←

=

all are used as per language.

Standard Notations and common functions



Input and output

Input: Read Variables names (scanf)

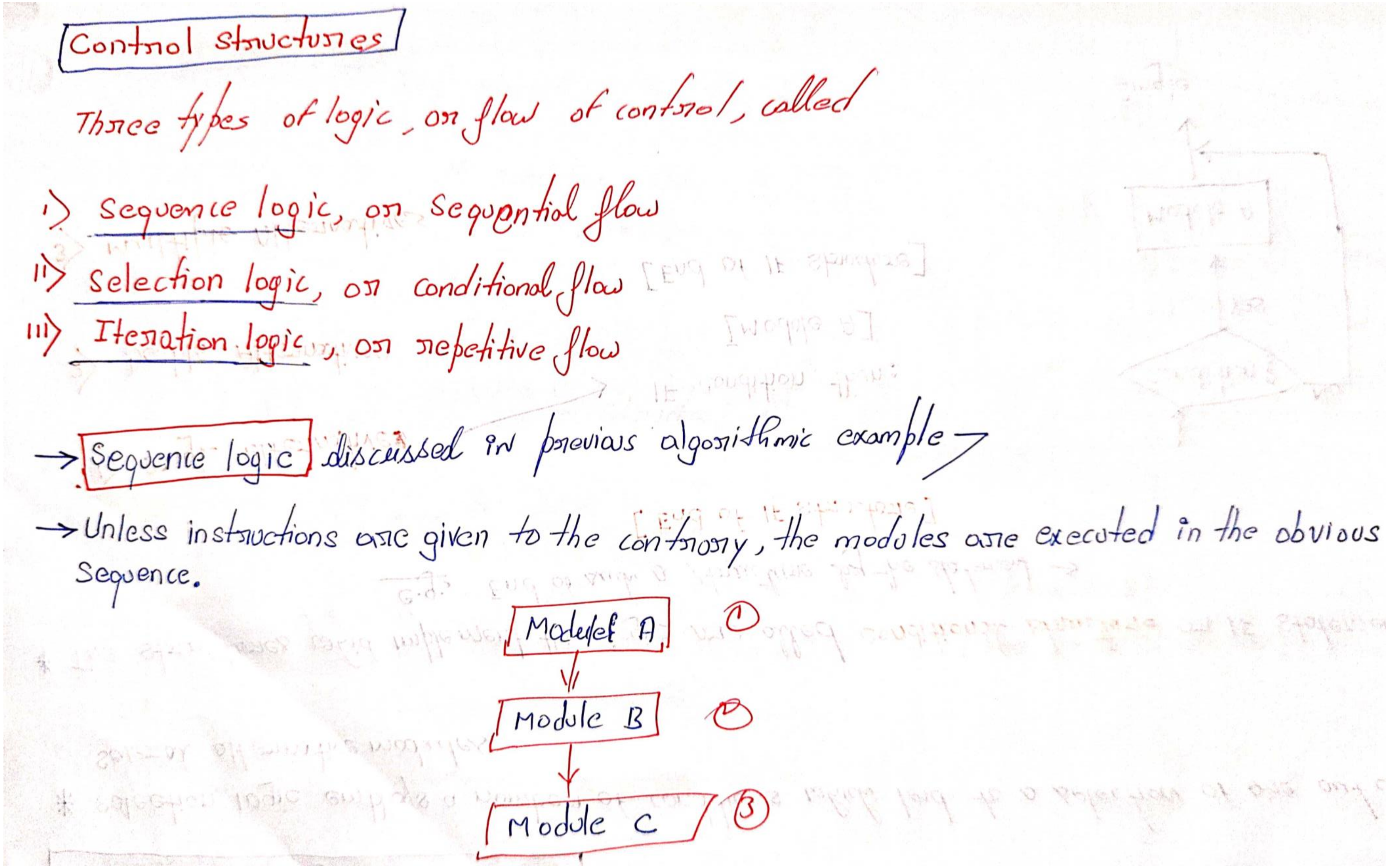
Output: Write Messages and/or variable names (printf)

Procedures

→ Used for an independent algorithmic module which solves a particular problem.

$\underbrace{\text{Procedures / Modules}}_{\text{specific}}$ / $\overbrace{\text{Algorithm}}^{\text{Broad.}}$ → Interchangable

Standard Notations and common functions



Control structures

Three types of logic, or flow of control, called

i) Sequence logic, or Sequential flow

ii) Selection logic, or conditional flow

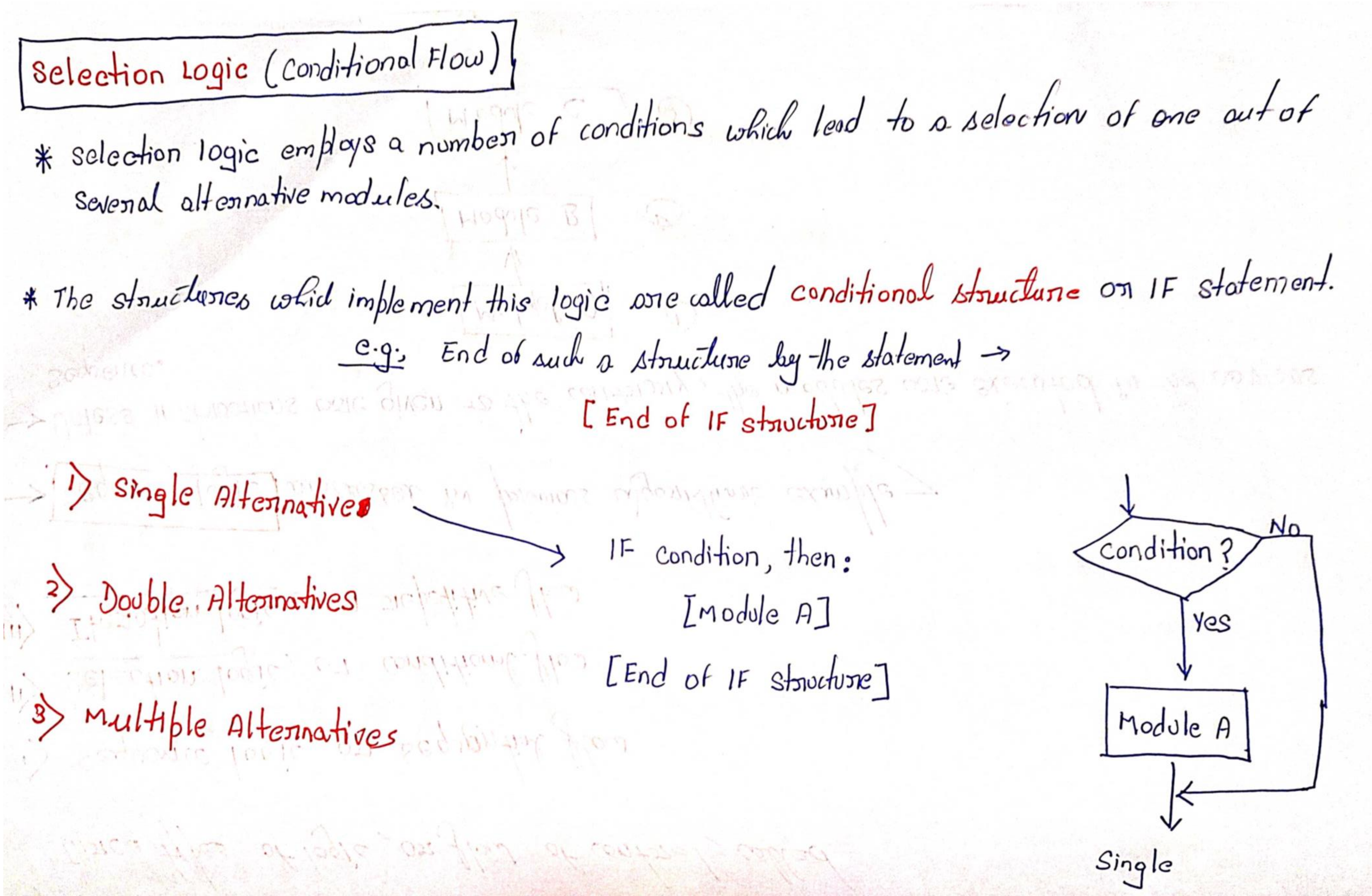
iii) Iteration logic, or repetitive flow

→ Sequence logic discussed in previous algorithmic example →

→ Unless instructions are given to the contrary, the modules are executed in the obvious sequence.

Module A
①
Module B
②
Module C
③

Standard Notations and common functions



Selection Logic (Conditional Flow)

* selection logic employs a number of conditions which lead to a selection of one out of several alternative modules.

* The structures whid implement this logic are called conditional structure or IF statement.

e.g.: End of such a structure by the statement →

[End of IF structure]

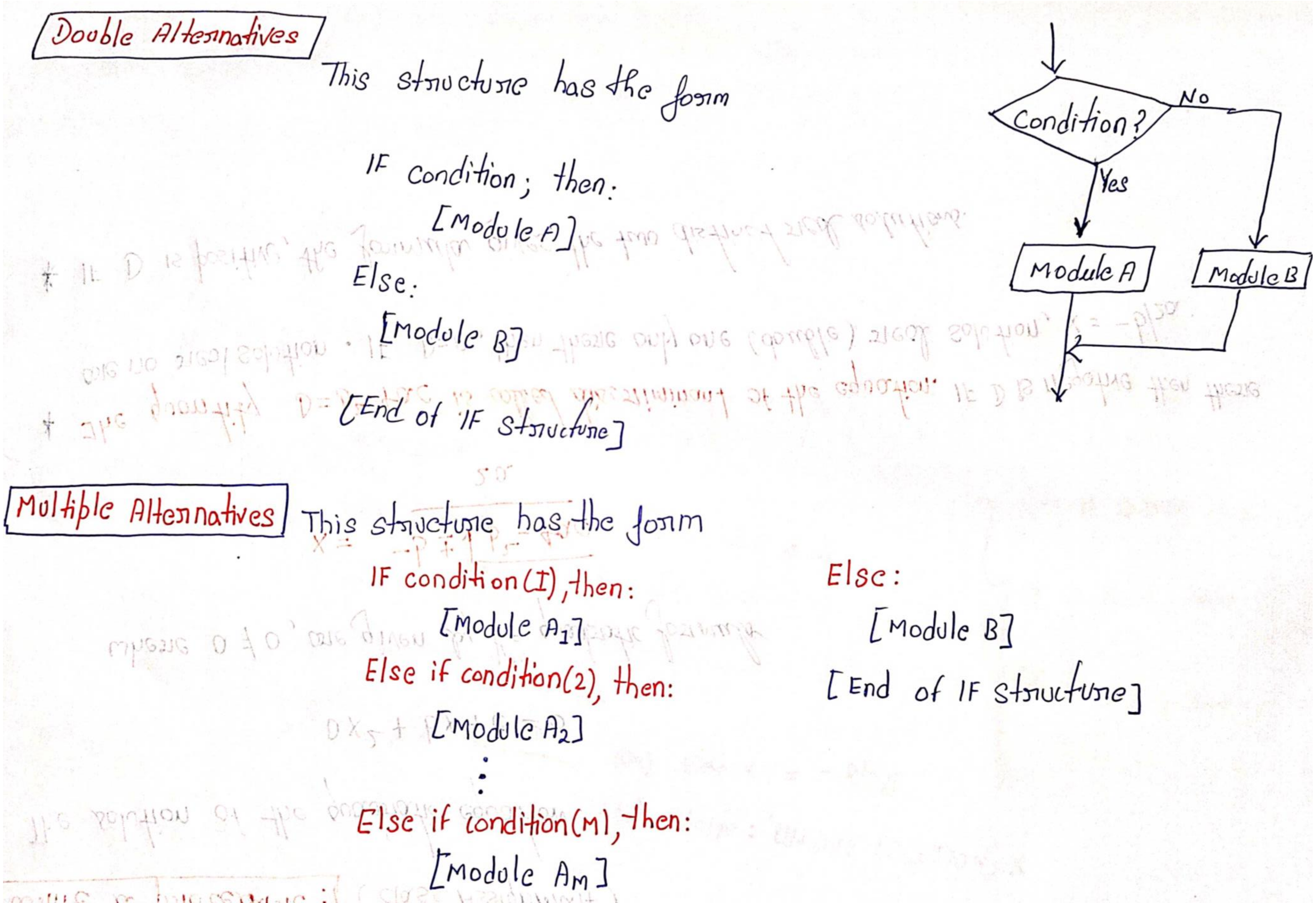
1> Single Alternatives

2> Double Alternatives

3> Multiple Alternatives

```
IF Condition, then:
        [Module A]
[End of IF Structure]
```

Standard Notations and common functions



Double Alternatives

This structure has the form

```
IF condition; then:
    [Module A]
Else:
    [Module B]
[End of IF structure]
```

Condition?
No
Yes
Module A
Module B

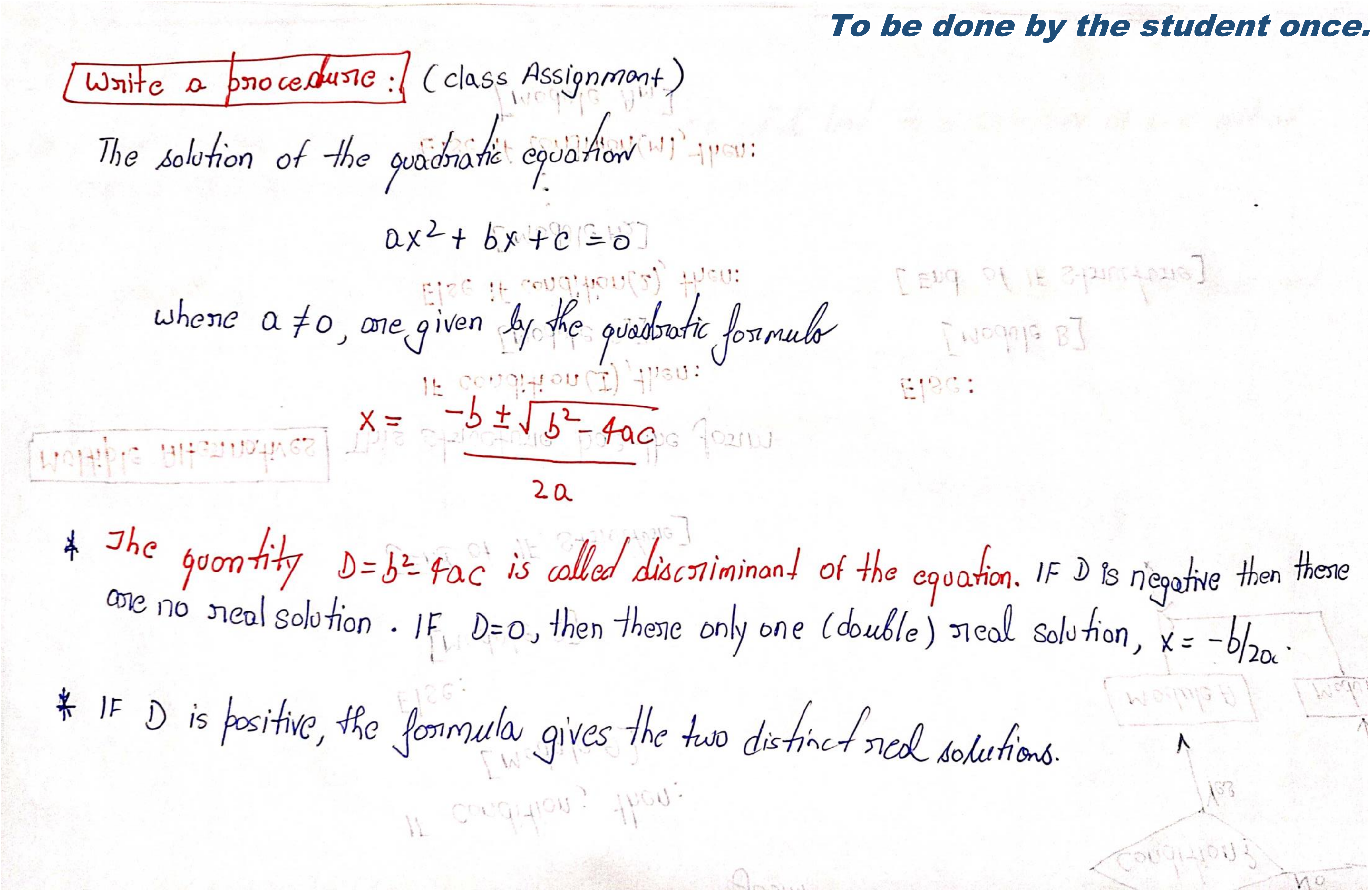
Multiple Alternatives

This structure has the form

```
IF condition(I), then:
    [Module A1]
Else if condition(2), then:
    [Module A2]
    :
Else if condition(M), then:
    [Module AM]
Else:
    [Module B]
[End of IF structure]
```

Standard Notations and common functions

To be done by the student once…



Write a procedure: (class Assignment)

The solution of the quadratic equation

$$ax^2 + bx + c = 0$$

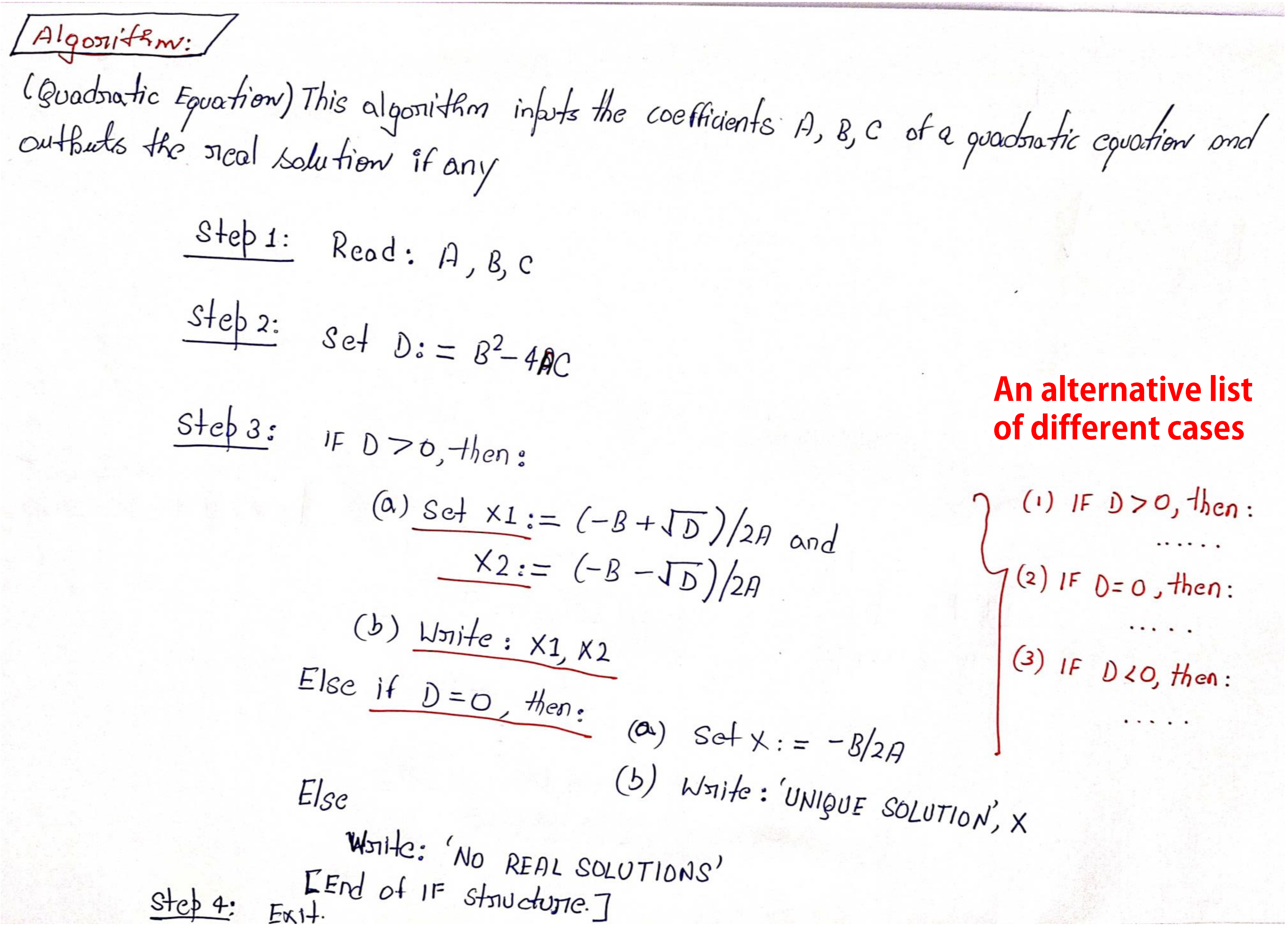
where $a \neq 0$, are given by the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

* The quantity $D = b^2 - 4ac$ is called discriminant of the equation. If D is negative then there are no real solution. If $D = 0$, then there only one (double) real solution, $x = -b/2a$.

* If D is positive, the formula gives the two distinct real solutions.

Standard Notations and common functions



Algorithm:

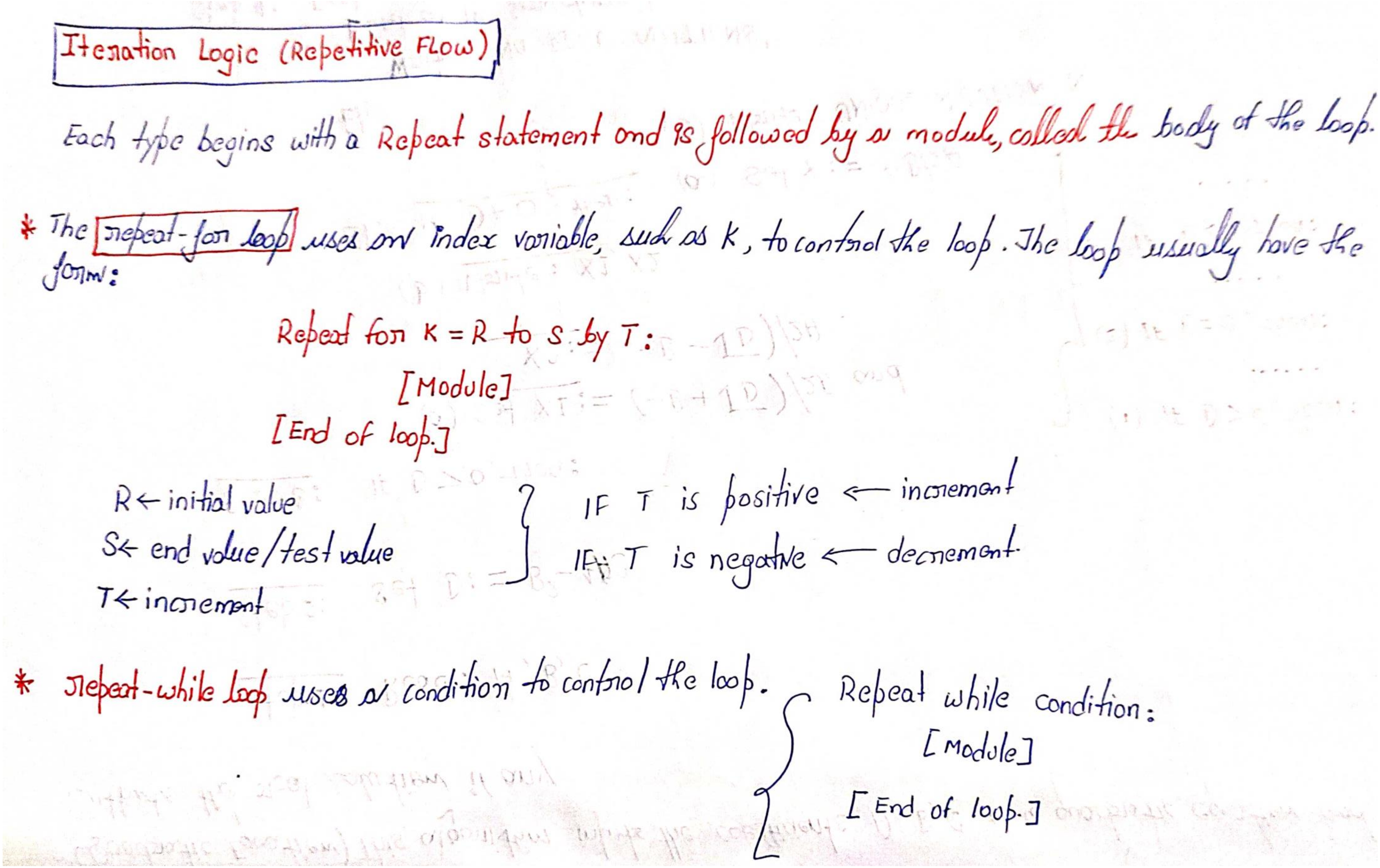
(Quadratic Equation) This algorithm inputs the coefficients A, B, C of a quadratic equation and outputs the real solution if any

```
Step 1:  Read: A, B, C
Step 2:  Set D := $B^2 - 4AC$
Step 3:  IF D > 0, then:
              (a) Set X1 := $(-B + \sqrt{D})/2A$ and
                  X2 := $(-B - \sqrt{D})/2A$
              (b) Write: X1, X2
         Else if D = 0, then:
              (a) Set X := $-B/2A$
              (b) Write: 'UNIQUE SOLUTION', X
         Else
              Write: 'NO REAL SOLUTIONS'
         [End of IF structure.]
Step 4:  Exit.
```

An alternative list of different cases

```
(1) IF D > 0, then:
        .....
(2) IF D = 0, then:
        .....
(3) IF D < 0, then:
        .....
```

Standard Notations and common functions



Iteration Logic (Repetitive Flow)

Each type begins with a Repeat statement and is followed by a module, called the body of the loop.

* The repeat-for loop uses an index variable, such as K, to control the loop. The loop usually have the form:

```
Repeat for K = R to S by T:
        [Module]
[End of loop.]
```

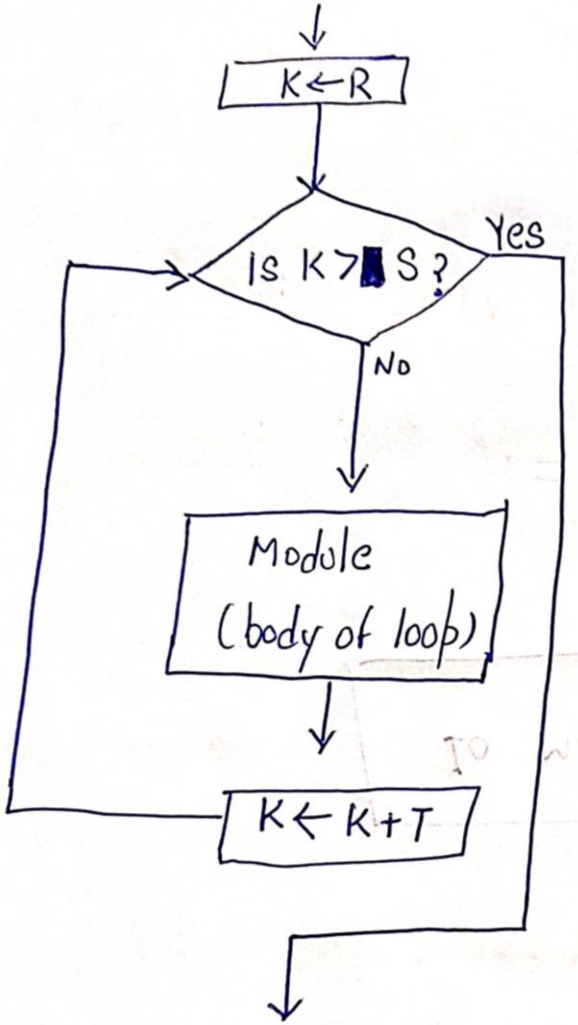
R← initial value
S← end value/test value
T← increment

IF T is positive ← increment
IF T is negative ← decrement.

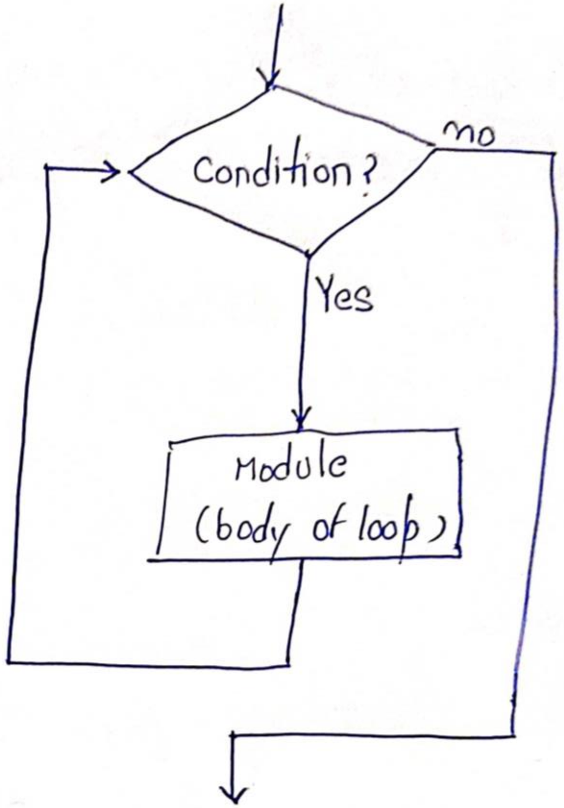
* repeat-while loop uses a condition to control the loop.

```
Repeat while condition:
        [Module]
[End of loop.]
```

Standard Notations and common functions



ⓐ Repeat-For structure

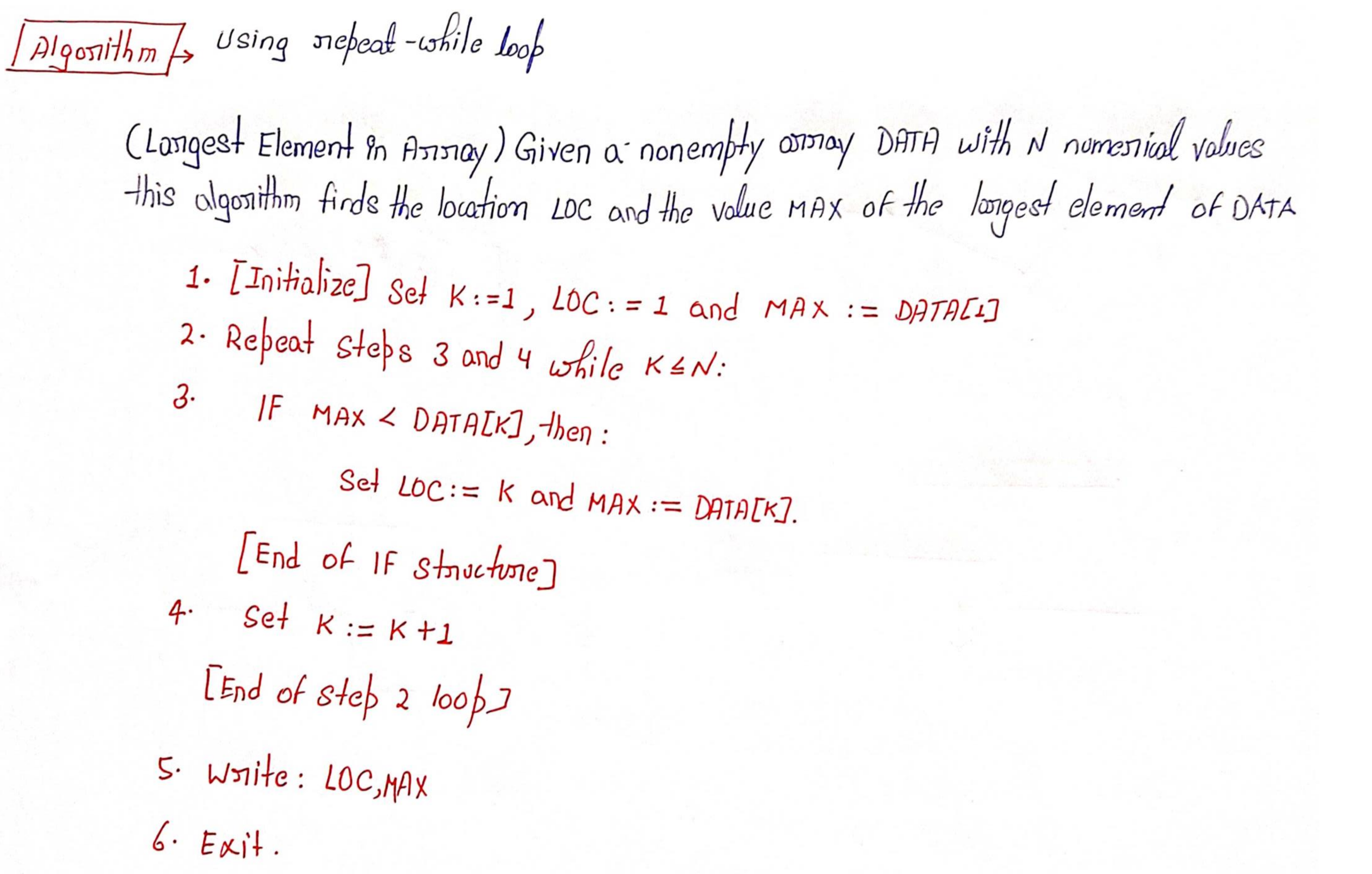


ⓑ Repeat-while structure

Rewrite the previous Algorithm using a repeat-while loop rather than a Goto statement.

10 minutes

Standard Notations and common functions



Algorithm → Using repeat-while loop

(Largest Element in Array) Given a nonempty array DATA with N numerical values this algorithm finds the location LOC and the value MAX of the largest element of DATA

```
1. [Initialize] Set K:=1, LOC:=1 and MAX:= DATA[1]
2. Repeat steps 3 and 4 while K≤N:
3.     IF MAX < DATA[K], then:
            Set LOC:= K and MAX:= DATA[K].
       [End of IF structure]
4.     Set K:= K+1
   [End of step 2 loop]
5. Write: LOC,MAX
6. Exit.
```

Algorithm Analysis and Runtime

Algorithmic analysis and runtime

- **Algorithm analysis** is the process of evaluating the performance of an algorithm in terms of its **efficiency and scalability**.

- The primary goal of algorithm analysis is to understand how an algorithm will behave as the size of the input data increases and to identify any bottlenecks or performance issues that may arise.

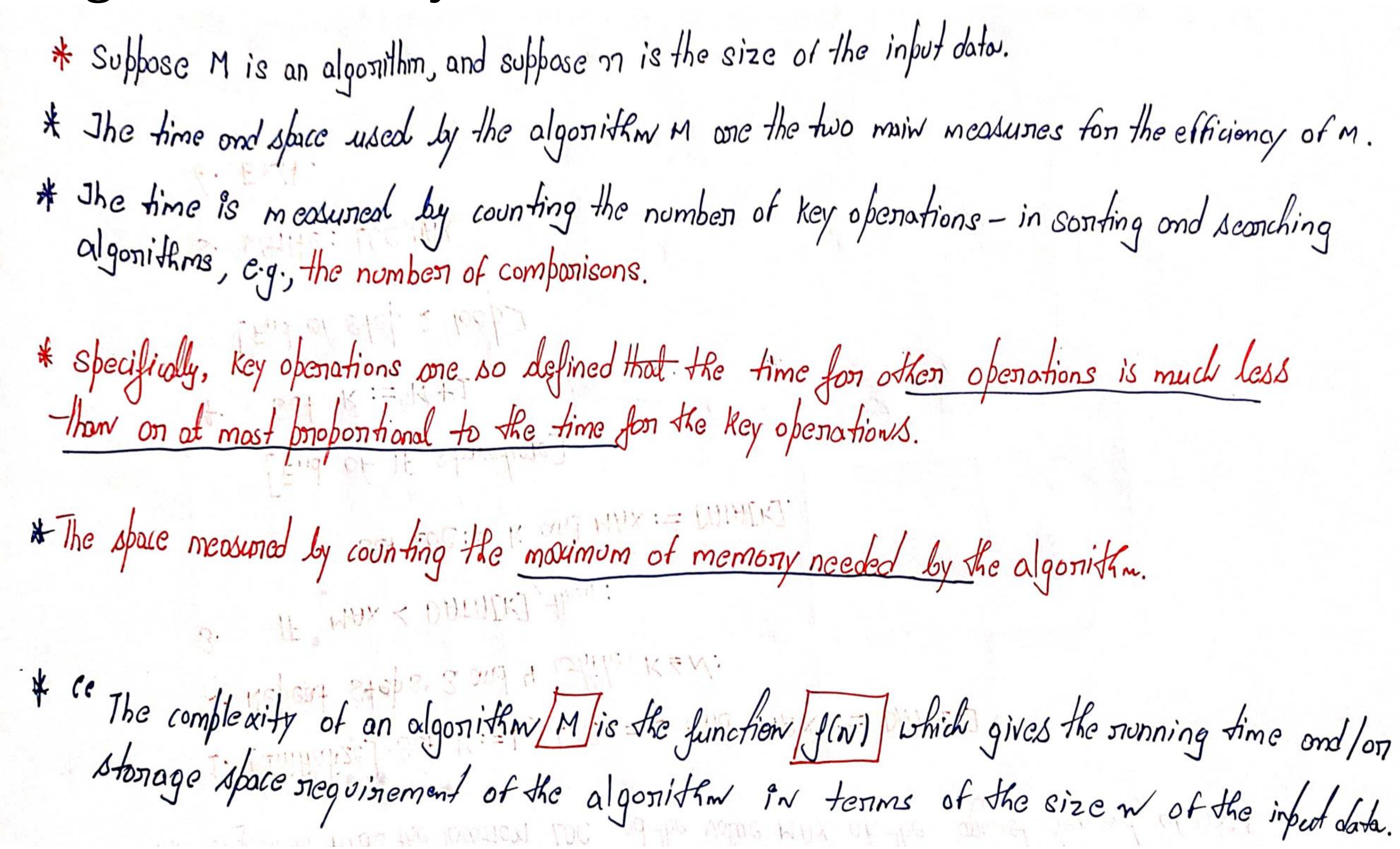
- One common approach to algorithm analysis is to measure the **running time of the algorithm as a function of the input size**. This can be done empirically by running the algorithm on inputs of different sizes and measuring the time it takes to complete each run.

- Alternatively, the time complexity of the algorithm can be analyzed theoretically, by analyzing the **number of operations the algorithm performs as a function of the input size**.

Algorithmic analysis and runtime

- Other factors that can affect the performance of an algorithm include the **use of memory**, **the use of parallelism**, and the **use of heuristics** or **other optimization techniques**.
- These factors can also be analyzed using algorithm analysis techniques, such as space complexity analysis, parallelism analysis, and optimization analysis.
- Algorithm analysis is a critical tool for understanding the performance of algorithms and for designing efficient algorithms for complex problems.
- It is widely used in computer science, engineering, and other fields to optimize performance and solve complex problems.

Algorithmic analysis and runtime



* Suppose M is an algorithm, and suppose n is the size of the input data.

* The time and space used by the algorithm M are the two main measures for the efficiency of M.

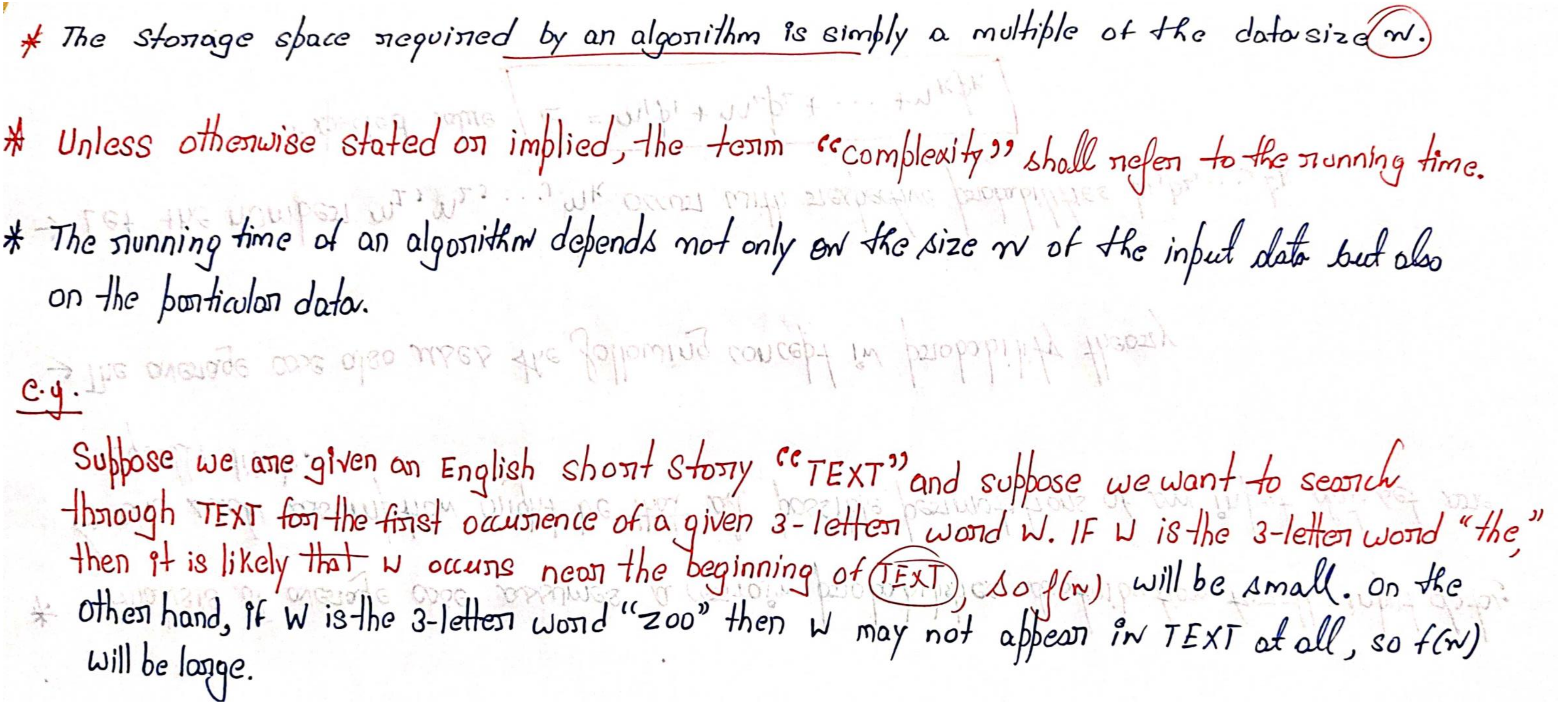
* The time is measured by counting the number of key operations – in sorting and searching algorithms, e.g., the number of comparisons.

* Specifically, key operations are so defined that the time for other operations is much less than or at most proportional to the time for the key operations.

* The space measured by counting the maximum of memory needed by the algorithm.

* " The complexity of an algorithm $\boxed{M}$ is the function $\boxed{f(n)}$ which gives the running time and/or storage space requirement of the algorithm in terms of the size n of the input data.

Algorithmic analysis and runtime



* The storage space required by an algorithm is simply a multiple of the data size n.

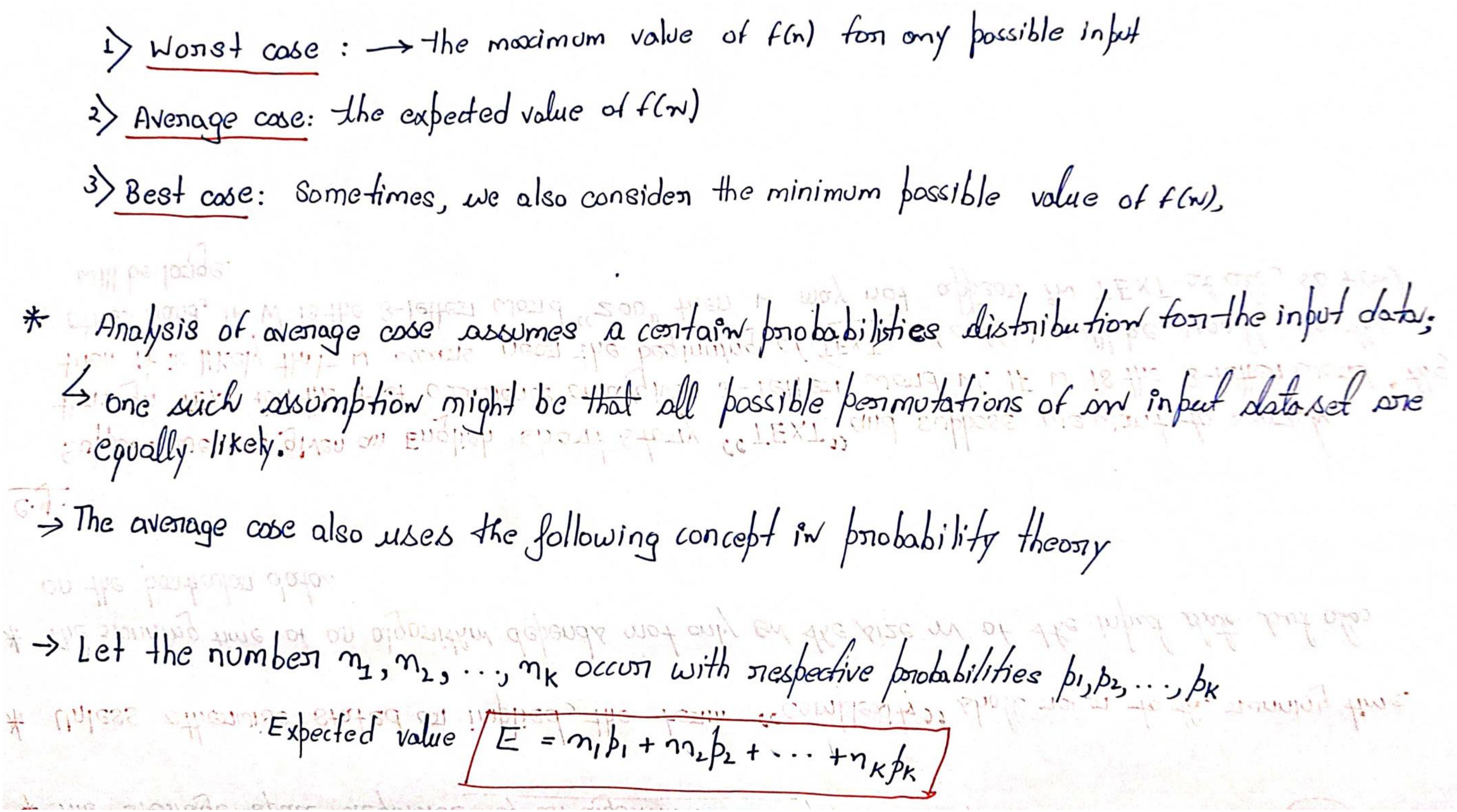
* Unless otherwise stated or implied, the term "complexity" shall refer to the running time.

* The running time of an algorithm depends not only on the size n of the input data but also on the particular data.

e.g.

Suppose we are given an English short story "TEXT" and suppose we want to search through TEXT for the first occurrence of a given 3-letter word W. If W is the 3-letter word "the," then it is likely that W occurs near the beginning of TEXT, so $f(n)$ will be small. On the other hand, if W is the 3-letter word "zoo" then W may not appear in TEXT at all, so $f(n)$ will be large.

Algorithmic analysis and runtime



1) Worst case : → the maximum value of $f(n)$ for any possible input

2) Average case: the expected value of $f(n)$

3) Best case: Sometimes, we also consider the minimum possible value of $f(n)$,

* Analysis of average case assumes a certain probabilities distribution for the input data;

↳ one such assumption might be that all possible permutations of an input dataset are equally likely.

→ The average case also uses the following concept in probability theory

→ Let the number $n_1, n_2, \dots, n_K$ occur with respective probabilities $p_1, p_2, \dots, p_K$

Expected value $\boxed{E = n_1 p_1 + n_2 p_2 + \dots + n_K p_K}$