1. Differentiate between DATA and INFORMATION??

ANSWER:

DATA : Data is a raw and unorganized fact that required to be processed to make it meaningful. Data can be simple at the same time unorganized unless it is organized. Generally, data comprises facts, observations, perceptions numbers, characters, symbols, image, etc.

Data is always interpreted, by a human or machine, to derive meaning. So, data is meaningless. Data contains numbers, statements, and characters in a raw form.

INFORMATION:

Information is a set of data which is processed in a meaningful way according to the given requirement. Information is processed, structured, or presented in a given context to make it meaningful and useful.

It is processed data which includes data that possess context, relevance, and purpose. It also involves manipulation of raw data.

Information assigns meaning and improves the reliability of the data. It helps to ensure undesirability and reduces uncertainty. So, when the data is transformed into information, it never has any useless details.

2. What do you mean by ABSTRACT DATA TYPE? Explain it with real world example.

ANSWER:

An abstract data type (**ADT**) is basically a logical description or a specification of components of the data and the operations that are allowed, that is independent of the implementation.

ADTs are a theoretical concept in computer science, used in the design and analysis of algorithms, data structures, and software systems, and do not correspond to specific features of computer languages.

There may be thousands of ways in which a given ADT can be implemented, even when the coding language remains constant. Any such implementation must comply with the content-wise and behavioral description of the ADT.

Example:

For example, integers are an ADT, defined as the values 0, 1, −1, 2, 2, ..., and by the operations of addition, subtraction, multiplication, and division, together with greater than, less than, etc. which are independent of how the integers are represented by the computer. Typically integers are represented in as binary numbers, most often as two's complement, but might be binary-coded decimal or in ones' complement, but the user is abstracted from the concrete choice of representation, and can simply use the data as integers.

3. Define Algorithm and explain Characteristics of Algorithm?
   ANSWER :
   Algorithm is a step by step procedure, which defines a set of instructions to be executed in certain order to get the desired output.
   An algorithm are generally analyzed on two factors − time and space. That is, how much execution time and how much extra space required by the algorithm.
   Following are the characteristics of Algorithm:
   Unambiguous − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
   Input − An algorithm should have 0 or more well-defined inputs.
   Output − An algorithm should have 1 or more well-defined outputs, and should match the desired output.
   Finiteness − Algorithms must terminate after a finite number of steps.
   Effectiveness- It is measured in terms of time and space.

   Characteristics of Algorithm:
   1)Input specified
   The input is the data to be transformed during the computation to produce the output.An algorithm should have 0 or more well-defined inputs.Input precision requires that you know

what kind of data, how much and what form the data should be.

2)Output specified:

The output is the data resulting from the computation (your intended result). An algorithm should have 1 or more well-defined outputs, and should match the desired output.Output precision also r4)Effectiveness

For an algorithm to be effective, it means that all those steps that are required to get to output must be feasible with the available resources.It should not contain any unnecessary and redundant steps which could make an algorithm ineffective.equires that you know what kind of data, how much and what form the output should be

3)Definiteness

Algorithms must specify every step and the order the steps must be taken in the process.Definiteness means specifying the sequence of operations for turning input into output. Algorithm should be clear and unambiguous.Details of each step must be also be spelled out (including how to handle errors).It should contain everything quantitative and not qualitative.

4)Effectiveness:

For an algorithm to be effective, it means that all those steps that are required to get to output must be feasible with the available resources.It should not contain any unnecessary and redundant steps which could make an algorithm ineffective.

5)Finiteness

The algorithm must stop, eventually.Stopping may mean that you get the expected output OR you get a response that no

solution is possible. Algorithms must terminate after a finite number of steps.An algorithm should not be infinite and always terminate after definite number of steps.

6)Independent
An algorithm should have step-by-step directions, which should be independent of any programming code.It should be such that it could be run on any of the programming languages.

4.Explain Big-O notation??

Ans:

When we compute the time complexity T(n) of an algorithm we rarely get an exact result, just an estimate. That’s fine, in computer science we are typically only interested in how fast T(n) is growing as a function of the input size n.

For example, if an algorithm increments each number in a list of length n, we might say: “This algorithm runs in O(n) time and performs O(1) work for each element”.
Big O notation is the language we use for talking about how long an algorithm takes to run. It's how we compare the efficiency of different approaches to a problem.

It's like math except it's an awesome, not-boring kind of math where you get to wave your hands through the details and just focus on what's basically happening.

With big O notation we express the runtime in terms of—brace yourself—how quickly it grows relative to the input, as the input gets arbitrarily large.

Q5. What are Guidelines for Asymptotic Analysis?
Ans:
Asymptotic analysis of an algorithm refers to defining the mathematical boundation/framing of its run-time performance. Using asymptotic analysis, we can very well conclude the best case, average case, and worst case scenario of an algorithm.

Asymptotic analysis is input bound i.e., if there's no input to the algorithm, it is concluded to work in a constant time. Other than the "input" all other factors are considered constant.

Asymptotic analysis refers to computing the running time of any operation in mathematical units of computation. For example, the running time of one operation is computed as f(n) and may be for another operation it is computed as g(n2). This means the first operation running time will increase linearly with the increase in n and the running time of the second operation will increase exponentially when n increases. Similarly, the running time of both operations will be nearly the same if n is significantly small.

Usually, the time required by an algorithm falls under three types −

Best Case − Minimum time required for program execution.

Average Case − Average time required for program execution.

Worst Case − Maximum time required for program execution.