ADT: we only consider the features/operations of the data type (from a logical/user view)

Stacks:

-e.g Hanoi Tower, stack of plates, pack of tennis balls.

-Follow the Last-in First-Out principle

-is a list with the restriction that insertion and deletion can be performed only from one end, called the top.

-Operations take constant time O(1)

->Invert their order

Queues:

-Insertion and deletion follow the FIFO principle (First-in, First-out)

-the item that has been in the queue for the longest time is deleted first

-insertion can be performed at one end (rear) --> enqueue

-deletion can be performed at the other end (front) --> dequeue

-principle of circular array: if all the elememnts are enqued at rear but there are 2 sposts left are front, if we enqueue we add elements in those spots

-if (rear + 1)%n == front then the list is full

-Operations take constant time O(1)

-> don't invert their order

-the position where the item has to be put in is equal t (equal to a counter, counts n of elements)

-e.g A = [10,9,8] count/t = 3 and the next element with enqueue has to be inserted at index 3.

-elements are inserted normally (if first we call enqueue(1) and enqueue(2) the array looks like A=[1,2])

-Dequeue: we move every single element from index h to index h+1.

Infix, Postfix, Prefix:

-Operands: Object on which the operation is executed

-Operators (e.g +, -, *, /)

e.g. 2 + 3: 2 and 3 are the operand and + is the operator

Infix: <Operand> <Operator> <Operand> (e.g. (2+3)*4, A-B, (p+2)*(r+5), a + b * c)

-human-readable

Order of precende of Operations:

1) Parentheses
2) Exponents
3) Multiplication and Division (left to right)
4) Addition and Subtraction (left to right)

Prefix:

-good for machines

<Operator> <Operand> <Operand> (e.g. (+ 2 3), - A B, +a(*bc))

Postfix:

-good for machines

-easiest to parse (less costs in time and space when evaluating the expression)

<Operand> <Operand> <Operator> (e.g. (2 3 +), A B -, abc*+)