Notion of a Data Structure

The problem solving process can be described as follows.

MODEL	ADT Abstract Data Type	DS Data Structure
Idea and an Informal Algorithm	Algorithm/ Pseudo program	Program in C, Java, etc.
	Data Structure Independent / Language Independent	Data structure dependent / Language dependent
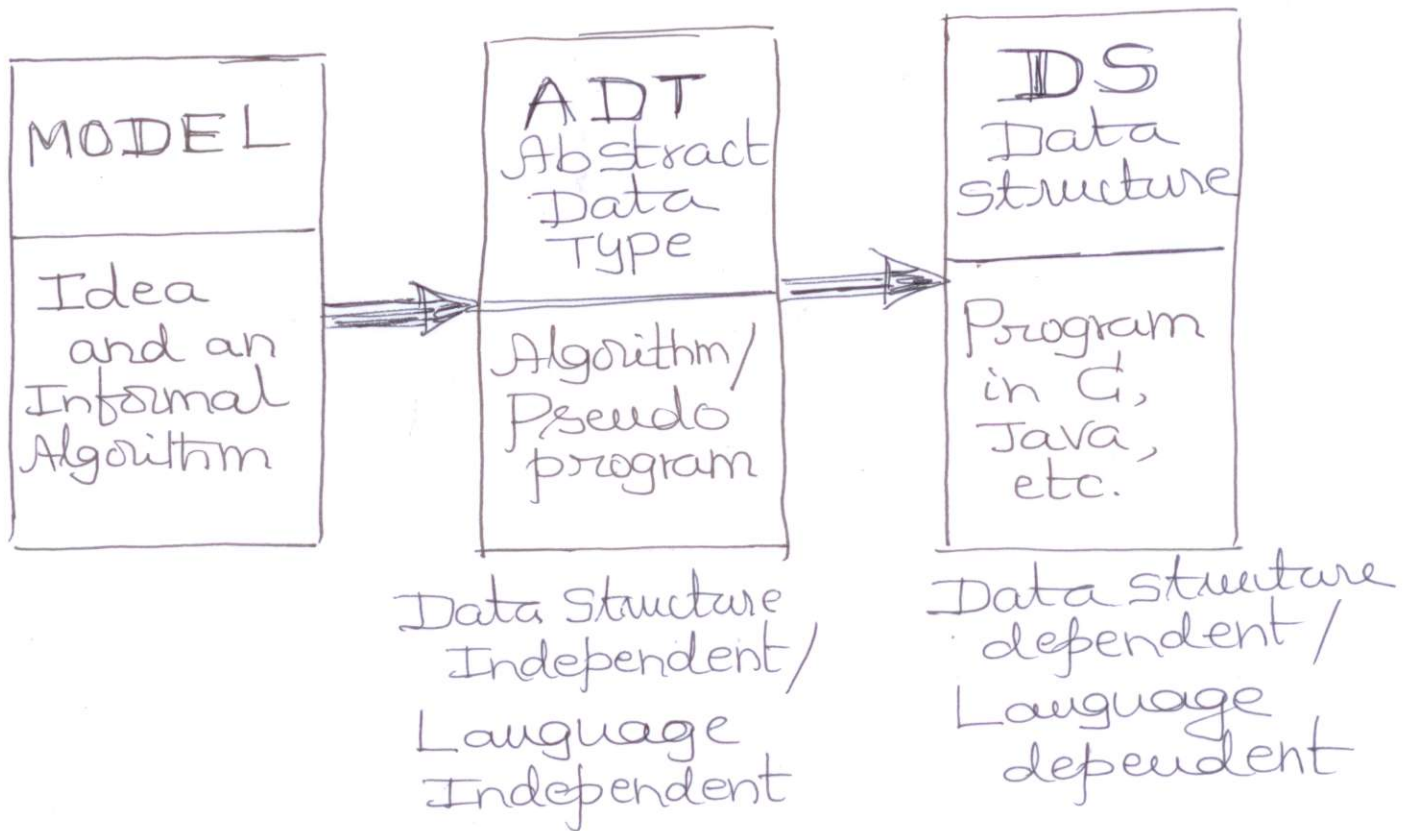
ADT	A set of elements together with a collection of well defined operations

DS	An implementation of an ADT in a programming language using data structuring facilities in the language

Example of ADT : List

A list, also called a sequence, is a collection of elements in a certain linear order.

$$a_1, a_2, \ldots, a_n \quad (n \geqslant 0)$$

• All the elements are of a particular type. For example, integers; strings; records; etc.

n — length of the list.

$n = 0$ — empty list

a_1 — first element

a_n — last element

a_i <u>precedes</u> a_{i+1}

a_i <u>succeeds</u> a_{i-1}

a_i is at position i

Given a list L, suppose we denote the last position in the list by end (L). Let p represent any position and x represent a typical element.

We can define operations such as

- insert (x, p, L)
- locate (x, L)
- retrieve (p, L)
- delete (p, L)
- next (p, L)
- prev (p, L)
- makenull (L)
- first (L)
- printlist (L)

⋮

A list of elements along with these operations is an abstract data type.

Pseudoprogram for eliminating duplicates in a list

Input : A given list L
Output : The list L with all duplicates removed

```
{ p = first (L);
  while (p != end (L))
  {
    q = next (p, L);
    while (q != end (L))
    { if (retrieve (p, L) == retrieve (q, L))
        delete (q, L);
      else
        q = next (q, L);
    }
    p = next (p, L)
  }
}
```
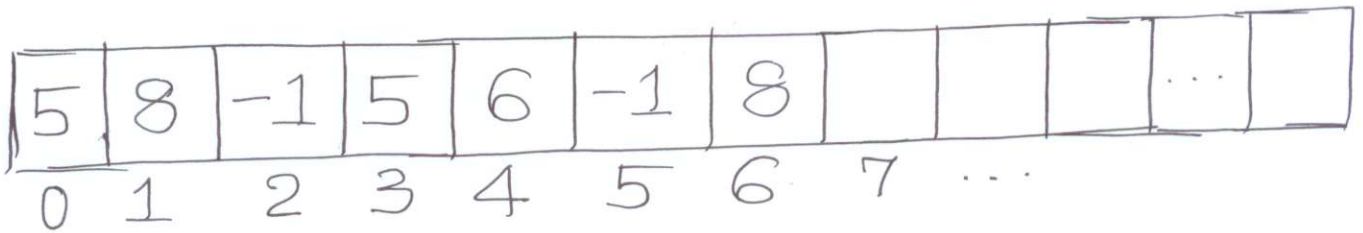
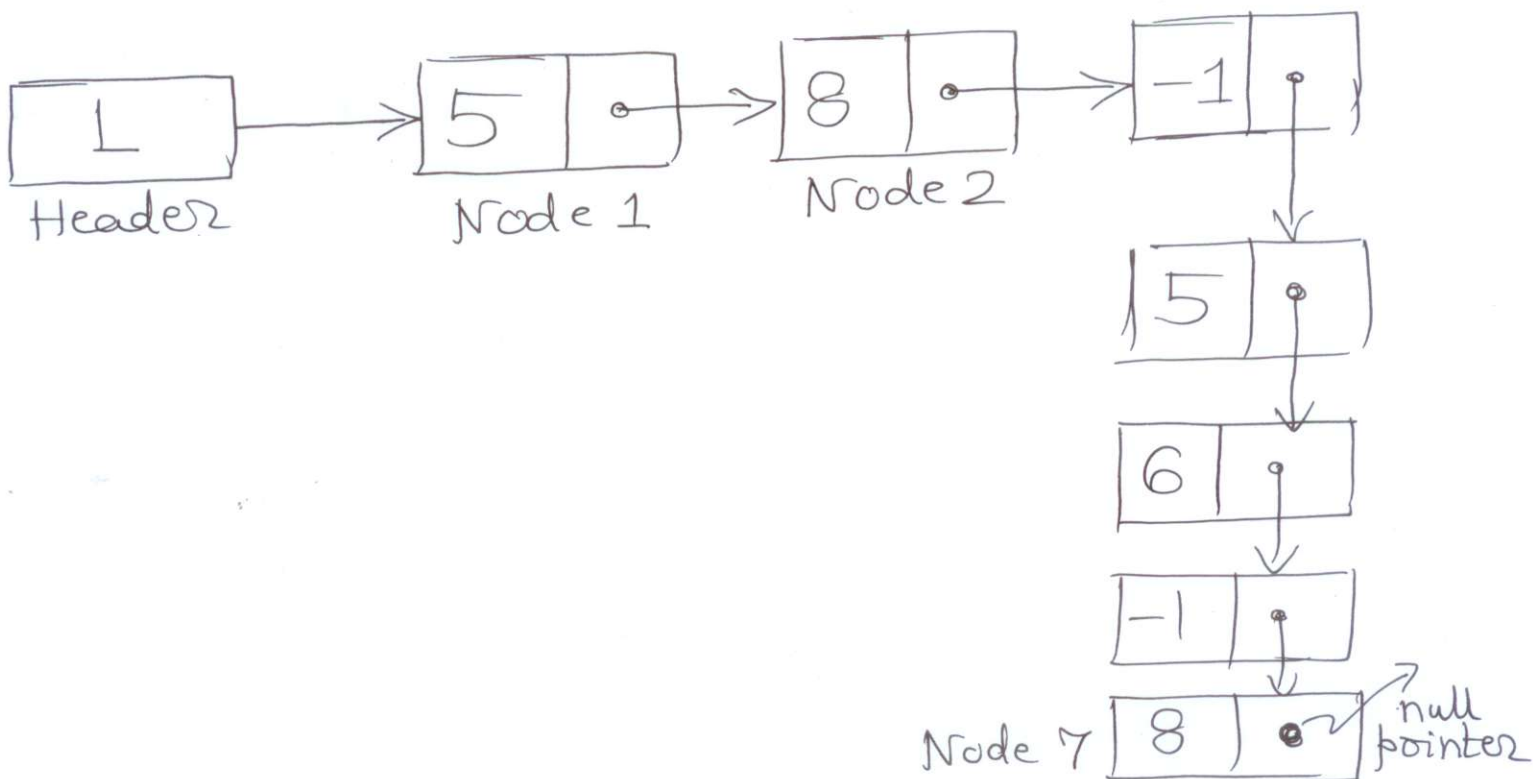The above code is independent of data structures and programming language

A list can be implemented using different data structures. Two major data structures are

— Arrays
— Linked Lists

Suppose the list is 5, 8, -1, 5, 6, -1, 8. The array representation would be

5	8	-1	5	6	-1	8				...	
0	1	2	3	4	5	6	7	...			

The linked list representation would be

L
Header

| 5 | • |
Node 1

| 8 | • |
Node 2

| -1 | • |

| 5 | • |

| 6 | • |

| -1 | • |

| 8 | • | → null pointer
Node 7

There are Tradeoffs involved in using arrays and linked lists:

- Arrays are random access while linked lists are sequential access

- Arrays have to be chosen to be of a certain maximum size leading to wastage of space

- Linked lists can grow or shrink according to our need. If the address is known, insertions and deletions are straightforward

- Linked lists use pointers which require additional space

Whether arrays are better or linked lists are better would depend on the set of operations to be implemented