ADT Set

"A set is a collection of members (or elements); each member of a set either is itself a set or is a primitive element called an atom. **All members of a set are different**, which means no set can contain two copies of the same element." //Source: Data Structures and Algorithm by Aho, Hopcroft, and Ullman

The set can be represented by a pair of curly braces, and the elements are between the pair. {3, 1, 5} is a set with member 1, 3, 5. Representations of Sets in memory can be the same as that of list but sets are different from list. Elements of the set are unique, i.e. no 2 or more copies of the same element. The order of the elements in the set is not significant.

ADT's based on Set

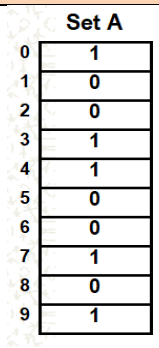
1. **ADT UID**
2. **Dictionary**
3. **Priority Queue**

ADT UID

- **Set ADT with operations**
  1. set **Union**,
  2. set **Intersection**, and
  3. set **Difference**
- ADT UID Implementations
  1. **Array – static or dynamic**
  2. **Linked List – singly or doubly linked, sorted or unsorted**
  3. **Cursor based**
  4. **Bit-Vector Implementation**

Practice Exercise 1: Write an appropriate definition of datatype Set implemented using linked list and write the code of the function setUnion(). Given 2 sets the function will return a set representing the union of the two given sets. Create 2 versions of the function setUnion(). In function setUnionV1(), assume that the sets are unsorted, while function setUnionV2(), the sets are sorted in ascending order. Determine the difference between the two functions' running time. How does being a sorted set affect the running time?

Bit-Vector Implementation

In Mathematics	Bit-Vector	Facts about Bit-vector Implementation of ADT set
//Universal Set U **U = { 0, 1, 2, 3, 4, 5, 6, 7 }** **Set A = {3, 7, 0, 4, 9}**		• A 1 dimensional array of bits (0, 1) or their equivalents in the given language • Elements of the Universal set are used as indices of the array • The C language does not support "bit data type", hence an array of int or char can be used to represent bit-vector • If Set A is implemented using bit-vector, then A[**x**] = 1 if **x** ∈ A A[**x**] = 0 if **x** ∉ A

- **Advantages of the Bit-Vector Implementation**
  1) Operations Member, Insert and Delete can be performed in constant time, **O(1)**
  2) Operations Union, Intersection and Difference can be performed in time proportional to the cardinality (no. of elements) of the universal set, **O(N)**
  3) If the Universal set is sufficiently small so that a bit vector fits in one computer word, then Union, Intersection and Difference can be performed by a single logical operation.
- **Disadvantage of the Bit-Vector Implementation**
  - **The amount of space needed for the set is proportional to the size of the universal set, hence if the universal set is relatively big then the size of each set (small or big) will also be big.**
  - **Illustration: Given the Set A = {3, 7, 0, 4, 9}**

A	
0	1
1	0
2	0
3	1
4	1
5	0
6	0
7	1
8	0
9	1

Q: What will happen if 7 is replaced by 77? How does it affect the size of the array representing the set?

Ans: ???

Conclusion: Bit vector implementation is best used when the universal set U is small, and its elements are integers (or can be mapped to the set of integers).

Practice Exercise 2:

Practice Exercise 2:

1. Write an appropriate definition of *datatype* **SET** such that in the declaration **SET A;** A is an array of integers of size 10.
   Note: SET A is a static array.
2. Using your definition of **SET** in #1, do the following
   1. Write the **function header of function Union()**. The function will return a new set C containing elements found in the given 2 sets A and B.
   2. Write an appropriate **function call**, **declare** and **initialize** the variables used in the call, BEFORE the call.
   3. Assume that the function call is in main(), **draw the execution stack**. Label variables with names, values and addresses.
   4. Write the **code of the function**
   5. **Simulate code** and **update** the execution stack.

SET A	
0	1
1	0
2	0
3	1
4	1
5	0
6	0
7	1
8	0
9	1