2. Composite Data Types

Define composite data type:

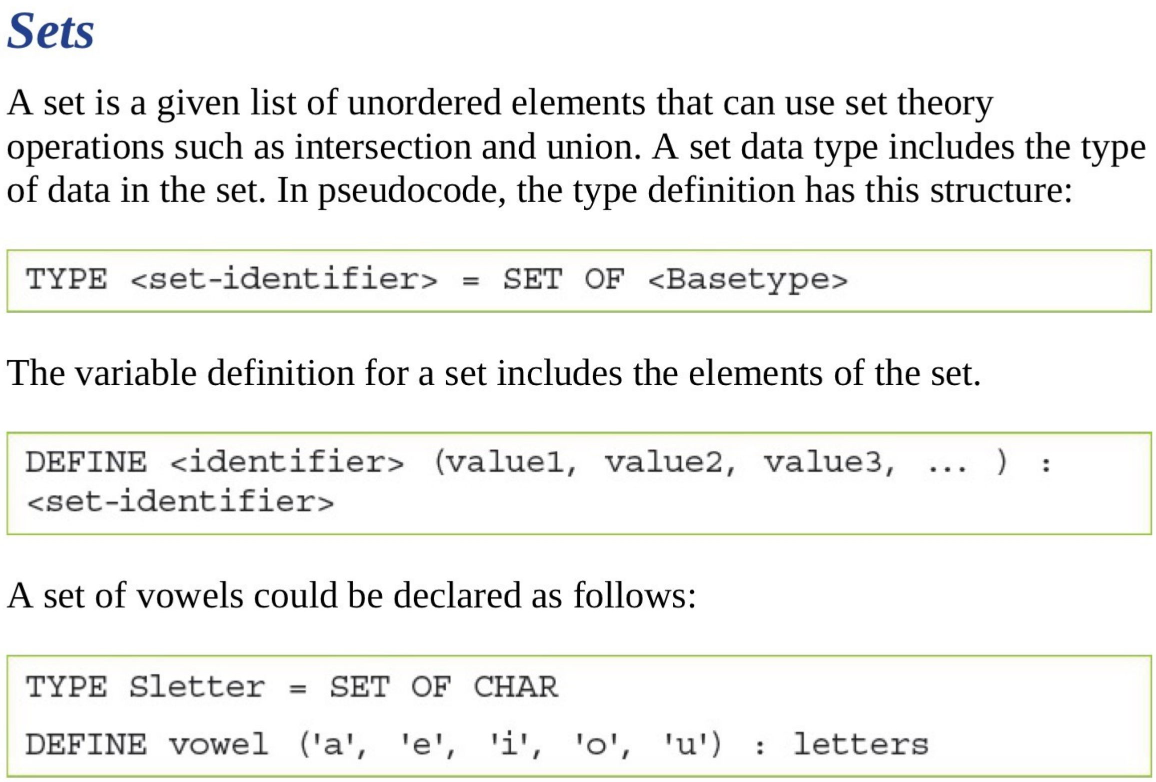
- Data type constructed from other data types

Give examples of composite data types:

- Array
  - Indexed collection of items with the same data type
- List
  - Indexed collection of items that can have different data type.
- Record
  - Collection of related items which may have different data types
- Set
  - Stores a finite numbers of different values that have no order // Support mathematical operation
- Class
- Stack
- Queue
- Linked List
- Dictionary

Set 🔄

> A data type to create sets and apply the mathematical operations defined in set theory
>
> - Contains a collection of data values
> - There is no organization of data values within the set
> - Duplicate values are now allowed



Sets

A set is a given list of unordered elements that can use set theory operations such as intersection and union. A set data type includes the type of data in the set. In pseudocode, the type definition has this structure:

```
TYPE <set-identifier> = SET OF <Basetype>
```

The variable definition for a set includes the elements of the set.

```
DEFINE <identifier> (value1, value2, value3, ... ) :
<set-identifier>
```

A set of vowels could be declared as follows:

```
TYPE Sletter = SET OF CHAR
DEFINE vowel ('a', 'e', 'i', 'o', 'u') : letters
```

```
TYPE letter = SET OF CHAR
DECLARE myname ('s','a','m','u','e','l'): letter
```

Supported operations:

- Check if a value exists in a set
- Adding a new data in set
- Delete a data in set
- Union two sets

Record ⏺

```
TYPE TbookRecord
  DECLARE title: STRING
  DECLARE author: STRING
  DECLARE publisher: STRING
  DECLARE noPages: STRING
  DECLARE isFiction: TRUE
ENDTYPE
```

Class/object🏛

> A template from which all objects are based
> Gives the properties and methods for an object

- Attributes: Properties/fields of an object
- Methods: Modules attached to objects to allow it to perform certain actions