1.6 SETS

DEF: A ***set*** is a collection of objects. The objects are called ***elements*** or ***members*** of the set.

NOTATION: : $x \in S$

Example 1.6.1:

$2 \in \{5, -7, \pi, \text{“algebra”}, 2, 2.718\}$
$8 \notin \{p : p \text{ is a prime number}\}$

SOME STANDARD SETS of NUMBERS

$\mathcal{N}$ = the set of all non-negative integers (the “natural numbers”)

$\mathcal{Z}$ = the set of all integers

$\mathcal{Z}^+$ = the set of all positive integers

$\mathcal{Q}$ = the rationals = $\{\frac{p}{q} : p, q \in \mathcal{Z} \text{ and } q \neq 0\}$

$\mathcal{R}$ = the real numbers

$\mathcal{C}$ = the complex numbers

ROSTERS for SETS

DEF: A ***roster*** specifies a finite set by enclosing in braces a list of representations of its elements. Repetitions and orderings are irrelevant to the content.

Example 1.6.2: a roster

$\{5, -7, \pi, \text{“algebra”}, 2, 2.718\}$

Example 1.6.3: identical sets

$\{1, 2, 3, 1, 1, 3\} = \{1, 2, 3\} = \{3, 1, 2\}$

DEF: The ***empty set*** is the set $\{\ \}$ having no elements. NOTATION: $\emptyset$.

Remark: In mathematics, there is only one empty set. However, a computer programming language may have a different empty set for every datatype.

Example 1.6.4: The empty set of character strings is equal to the empty set of lions.

DEF: A ***singleton set*** is a set with one element.

Example 1.6.5: $\{x\}$ is a singleton set.

SPECIFICATION by PREDICATES

A predicate over a well-defined set can specify any subcollection within that set. (Warning: This "set-builder" method can lead to non-sets.)

Example 1.6.6: $\{x \in \mathcal{Z} : P(x)\}$ where $P(x)$ is TRUE if x is prime.

Example 1.6.7: $\{(x, y) : x, y \in \mathcal{R} \wedge x^2 + y^2 = 1\}$

OTHER WAYS to SPECIFY SETS

(1) By prose. (can also lead to non-sets).

Example 1.6.8: The set of all palindromes.

(2) By operations on other sets.
Examples soon.

(3) By recursive construction.
Examples in §3.4.

SETS as ELEMENTS of SETS

An object x is not the same as the singleton set $\{x\}$. Moreover, $\{x\} \neq \{\{x\}\}$.

Analogy: Iterative pointers to a computer object creates new objects.
$x \neq \&x \neq \&\&x$

Analogy: Iterative enquotation of a character string creates new objects.
““lion”” ≠ “lion” ≠ lion

RELATIONS on SETS

DEF: Set X is a ***subset*** of set Y if every element of X is also an element of Y. NOTATION: $X \subseteq Y$.

DEF: A subset X of a set Y is ***proper*** if Y has at least one element that is not in X.

DEF: Sets X and Y are ***equal*** if each set is a subset of the other. NOTATION: $X = Y$.

Example 1.6.9:

(1) $\emptyset$ is a proper subset of every set except itself.

(2) The integers are a subset of the real numbers.

DISAMBIGUATION: In a computer programming languages in which the integers and the reals are distinct ***datatypes***, the integers are not a subset of the reals.

Remark: Whereas mathematics deals with objects, computation science deals with their representations.

POWER SET

DEF: The ***power set*** of a set S is the set of all subsets of S. NOTATION: 2^S or $P(S)$.

Example 1.6.10:

$P(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$.

$P(\emptyset) = \{\emptyset\}$.

$P(P(\emptyset)) = \{\emptyset, \{\emptyset\}\}$.

Proposition 1.6.1. *If set S has n elements, then the power set $P(S)$ has 2^n elements.* $\diamondsuit$

CARTESIAN PRODUCT

DEF: The ***cartesian product*** of sets A and B is the set $\{(a, b) \mid a \in A \wedge b \in B\}$. NOTATION: $A \times B$.

Example 1.6.11: $A = \{1, 2\}$ $B = \{a, b, c\} \Rightarrow A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$.

Proposition 1.6.2. *The cartesian product $A \times B$ is empty iff either A or B is empty.* $\diamondsuit$