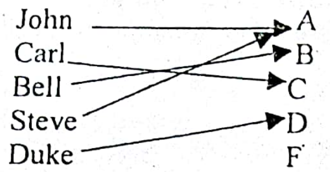
FUNCTIONS

Definition: Let A and B be sets. If there is a rule f such that under this rule every element of set A is assigned to one and one element of B, then the rule f is called a function from A to B. It is denoted by $f: A \to B$ which is read: "f is a function of A to B" or "f maps A to B".

Ex: Suppose that each student in a discrete mathematics class is assigned a letter grade from the set $\{A, B, C, D, F\}$. And suppose that the grades are A for John, C for Carl, B for Bell, D for Duke, A for Steve. This assignment of grade is shown in the figure.



if f is a function from A to B, we say that f maps A to B.

Domain and Codomain:

If f is a function from A to B i.e. $f: A \to B$ then A is called the domain of function f and B is called the codomain of function f.

Image of a function:

Let f is a function from A to B i,e, $f: A \to B$. If $a \in A$ then the element of B which is assigned to a is called the image of a and denoted by $f(a)$.

- a is also called the pre image of b where $b \in B$.

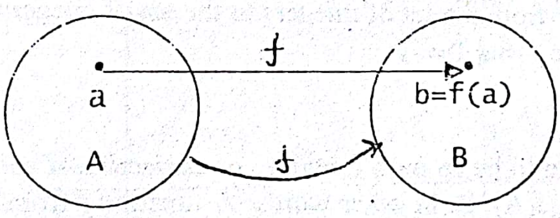


Fig: Representation of a function f from A to B.

Range of a function:

Let $f: A \to B$ be a function, then range of f is the set of those elements of B which are the images of some elements of A. In other words, we can say that range of a function f is the set of all images of elements of A.

- We denote the range of $f: A \to B$ by $f(A)$.
- $f(A)$ is a subset of B i,e, $f(A) \subseteq B$.

Ex: Let f be a function from **Z** to **Z** that assigns the square of an integer to this integer i.e. $f: x \to x^2$. Then the domain of the f is the set of all integers, the codomain of f can be chosen to be the set of integers, and the range of f is the set of all nonnegative integers that are perfect squares, namely $\{0, 1, 4, 9, \ldots\}$.

- A function can be expressed by means of a mathematical formula. If $f: x \to 2x$ then we may describe this function by writing $f(x) = 2x$ or $y = 2x$.

In the first notation x is called a *variable* and the letter f denotes the function. In the second notation, x is called the *independent variable* and y is called the *dependent variable* since the value of y will depend on the value of x.

- The domain and codomain of functions are often specified in programming languages. For instance, the Pascal statement

Function floor (x: real): integer

States that the domain of the floor function is the set of real numbers and its codomain is the set of integers.

- Two real valued functions with the same domain can be added and multiplied. Let f_1 and f_2 be function from A to **R**. then $f_1 + f_2$ and $f_1 f_2$ are also functions from **A** to **R**.

$$(f_1 + f_2)(x) = f_1(x) + f_2(x)$$
$$(f_1 * f_2)(x) = f_1(x) f_2(x)$$

Ex: Let f_1 and f_2 be functions from **R** to **R** such that $f_1(x) = x^2$ and $f_2(x) = x - x^2$. Then

$$(f_1 + f_2)(x) = x^2 + (x - x^2) = x$$
$$f_1 f_2(x) = x^2 (x - x^2) = x^3 - x^4$$

One-to-One Function:

A function f: A→ B is said to be one-to-one (written 1-1) or ***injective*** if different elements in the domain A have distinct images. In other words, A function f is said to be one-to-one if and only if $f(x) = f(y)$ implies that x = y for all x and y in the domain of f.

Ex: The function f from $\{a, b, c, d\}$ to $\{1, 2, 3, 4, 5\}$ with $f(a) = 4$, $f(b) = 5$, $f(c) = 1$, $f(d) = 3$ is one-to-one.

Ex: The function $f(x) = x^2$ from the set of integers to the set of integers is not one-to-one. Because, for instance, $f(1) = f(-1) = 1$, but $1 \neq -1$.

Onto Function:

A function f: A→ B is said to be an onto function, or ***surjective,*** if each element of B is the image of some element of A. i,e, f(A)=B. In other words, A function f from A to B is called onto if and only if for every element b∈B there is an element a∈A with $f(a) = b$.

Ex: Let f be a function from $\{a, b, c, d\}$ to $\{1, 2, 3\}$ defined by $f(a) = 3$, $f(b) = 2$, $f(c) = 1$, $f(d) = 3$. Since all three element of the codomain are images of elements in the domain, we see that f is onto.

Ex: The function $f(x) = x^2$ from the set of integers to the set of integers is not onto, since there is no integer x with $x^2 = -1$, for instance.

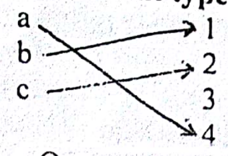
Invertible Functions:

A function f: A→B is *invertible* or ***bijection*** or ***one-to-one correspondence,*** if and only if it is both one-to-one and onto.

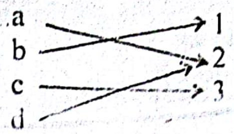
Ex: The function from $\{a, b, c, d\}$ to $\{1, 2, 3, 4\}$ with $f(a) = 4$, $f(b) = 2$, $f(c) = 1$, $f(d) = 3$ is one-to-one and onto. It is one-to-one since the function takes on distinct values. It is onto since all four elements of the codomain are images of elements in the domain. Hence, f is a bijection.

- It is called one-to-one correspondence since each element of A will correspondence to a unique element of B and vice versa.

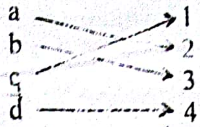
...amples of different types of correspondences:



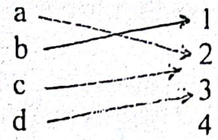
One-to-one
Not Onto



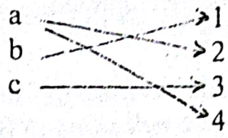
Not one-to-one
Onto



One-to-one
Onto



Not one-to-one
Not Onto



Not a function

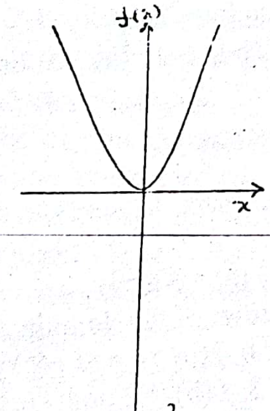
Geometrical characteristics of one-to-one and onto function:

A function f: A→B is one-to-one means that there are no two distinct pairs (a_1,b) and (a_2,b) in the graph of f. Hence, each horizontal line can intersect the graph of f at most one point.

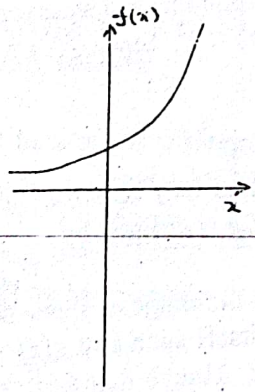
On the other hand, f is a onto function means that for every b∈B, there must be at least one a∈A such that (a,b) belongs to the graph of f. Hence, each horizontal line must intersect the graph of f at least once.

Accordingly, If f is both one-to-one and onto i.e. bijection then each horizontal line will intersect the graph of f in exactly one point.

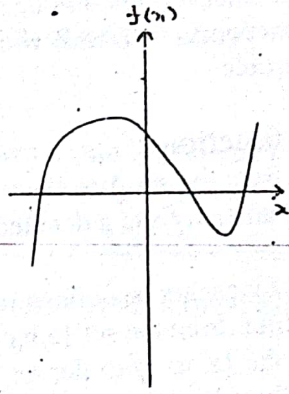
Ex: Consider the following four functions from R to R:



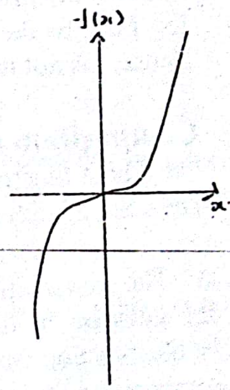
$f_1(x)=x^2$
Not one-to-one
Not onto



$f_2(x)=2^x$
One-to-one
Not onto



$f_3(x)=x^3-2x^2-5x+6$
Not one-to-one
Onto



$f_4(x)=x^3$
One-to-one
Onto

Identity function / Unit function:

Let A be a set. The identity function on A is the function f: A→A where $f(x) = x$ when x∈a. In other words, the identify function is the function that assigns each element to itself.

- Identity function is one-to-one and onto, so it is a bijection.

Constant function:

A function f: A→B is called a constant function if the same element b∈B is assigned to every element in A i.e. if the range of f consists of only one element.

Ex. $f(x) = 5$ is a constant function since the range of the function f is {5}.

ual functions:

If f and g are functions defined on the same domain D and if $f(a) = g(a)$ for every $a \in D$ then the functions f and g are equal and we write $f = g$.

Ex: $f: R \rightarrow R$ and $g: R \rightarrow R$

$f(x) = x^2$ $g(y) = y^2$

then $f = g$.

Ex: $f(x) = x^2$ x is real number

$g(x) = x^2$ x is complex number

$f \neq g$ (since different domain).

Inverse function:

Let f be a one-to-one correspondence from the set A to the set B. The inverse function of f is the function that assign to an element b belonging to B the unique element a in A such that $f(a) = b$.

- The inverse function of f is denoted by f^{-1}, hence $f^{-1}(b) = a$ when $f(a)=b$.

Why one-to-one correspondence?

- When not one-to-one some element ***b*** in the codomain is the image of more then one element in the domain. As a result, for inverse function we cannot assign to each element ***b*** in the domain a unique element ***a*** in the codomain.
- When not onto, for some element ***b*** in the codomain there are no element ***a*** in the domain exist. So, in inverse function for b (domain) there is no ***a*** (codomain).

Ex: Let f be function from $\{a, b, c\}$ to $\{1, 2, 3\}$ such that $f(a) = 2$, $f(b) = 3$, $f(c) = 1$. The function is invertible since it is a one-to-one correspondence. The inverse function f^{-1} reverses the correspondence given by f, so that $f^{-1}(1) = c$, $f^{-1}(2) = a$ and $f^{-1}(3) = b$.

Ex: Let f be the function from the set of integers to the set of integers such that $f(x) = x + 1$. The function has inverse since it is one-to-one correspondence. The inverse function $f^{-1}(y) = y-1$.

Ex: Let f be the function from Z to Z with $f(x) = x^2$. Since $f(1) = f(-1) = 1$, f is not one-to-one. Hence, f is not invertible.

Composition of functions:

Let g be a function from the set A to the set B and let f be a function from the set B to the set C. The composition of function f and g denoted by $f_o g$ is defined by

$$(f_o g)(a) = f(g(a))$$

- The composition $f_o g$ cannot be defined unless the range of g is a subset of the domain of f.

Ex: Let g be the function from the set {a,b,c} to itself such that $g(a) = b$, $g(b) = c$ and $g(c) = a$. Let f be the function from the set {a, b, c} to the set {1, 2, 3}such that $f(a) = 3$, $f(b) = 2$ and $f(c) = 1$. The composition $f_o g$ is defined by

$$(f_o g)(a) = f(g(a)) = f(b) = 2$$
$$(f_o g)(b) = f(g(b)) = f(c) = 1$$
$$(f_o g)(c) = f(g(c)) = f(a) = 3.$$

- $g_o f$ is not defined, because the range of f is not a subset of the domain of g.

Ex: Let f and g be the functions from the set of integers to the set of integers defined by $f(x)=2x+3$ and $g(x)=3x+2$. Both the compositions $f_o g$ and $g_o f$ are defined.

$$(f_o g)(x) = f(g(x)) = f(3x+2) = 2(3x+2)+3 = 6x+7$$
$$(g_o f)(x) = g(f(x)) = g(2x+3) = 3(2x+3)+2 = 6x+11$$

- Here $f_o g$ and $g_o f$ are not equal. So, the commutative law does not hold for the composition of functions.

aphs of functions:

Let f be a function from the set A to the set B. the graph of the function f is the set of ordered pairs $\{(a, b) | a \in A \text{ and } f(a)=b\}$.

Ex: The graph of the function $f(n) = 2n + 1$ from the set of integers to the set of integers is the set of ordered pairs of the form $(n, 2n+1)$ where n is an integer. Draw the graph.

Some Important Functions:

Floor and Ceiling functions:

Let x be any real numbers. Then x lies between two integers called the floor and the ceiling of x; specifically,

$\lfloor x \rfloor$, called the *floor* of x, denotes the greatest integers that does not exceed x.

$\lceil x \rceil$, called the *ceiling* of x, denotes the least integers that is not less then x.

Ex: $\lfloor 3.14 \rfloor = 3$ $\lfloor -8.5 \rfloor = -9$

$\lceil 3.14 \rceil = 4$ $\lceil -8.5 \rceil = -8$

- If x is an integer then $\lfloor x \rfloor = \lceil x \rceil$ otherwise $\lfloor x \rfloor + 1 = \lceil x \rceil$.

Integer value function:

Let x be any real number. The integer value of x written **INT** (x), converts x into an integer by deleting (or truncating) the fractional part of the number.

Ex. INT (3.14)=3 INT (-8.5)=-8

- When x is positive then INT (x) = $\lfloor x \rfloor$ and when x is negative then INT (x)= $\lceil x \rceil$.

Absolute value function:

The absolute value of the real number x, written **ABS** (x) or $|x|$, is defined as the greater of x or -x.

Ex: ABS (-15) = 15 ABS (15) = 15 ABS (0) = 0

- When x is positive then ABS (x) = x and when x is negative ABS (x) = -x.
- $|x| = |-x|$ and for $x \neq 0$, $|x|$ is positive.

Remainder function:

Let k be any integer and M be a positive integer. Then k (mod M) [read k modulo M] will denote the integer remainder when k is divided by M. More exactly, k (mod M) is the unique integer r such that $k = Mq + r$ where $0 \leq r < M$.

Ex: 25 (mod 7) = 4 25 (mod 5 = 0

- If k is negative, divide $|k|$ by M to obtain a remainder r′; then k (mod M) = M – r′ when $r' \neq 0$.

Ex: -26(mod 7) =7-5 =2

Exponential function:

Let m be a positive integer. Then $a^m = a.a.a.\ldots.a$ (m times)

- $a^0 = 1$, $a^{-m} = 1/a^m$, $a^{m/n} = \sqrt[n]{a^m} = (\sqrt[n]{a})^m$ (for any rational number m/n)
- $b^{x+y} = b^x . b^y$ and $(b^x)^y = b^{xy}$

Logarithmic function:

Let b be a positive number. The logarithm of any positive number x to be the base b, written $\log_b x$ represents the exponent to which b must be raised to obtain x. Thais $y = \log_b x$ and $b^y = x$ are equivalent statements.

Ex. $\log_2 8 = 3$ since $2^3 = 8$. $\log_b 1 = 0$ since $b^0 = 1$. $\log_b b = 1$ since $b^1 = b$

- Three classes of logarithms are of special importance: logarithms to base 10, called *common logarithms*($\log_{10} x$ or log x); logarithms to base e, called *natural logarithms*($\log_e x$ or ln x); and logarithms to base 2, called *binary logarithms*($\log_2 x$ or lg x or Log x).
- $\lfloor \log_2 100 \rfloor = 6$ and $\lceil \log_2 1000 \rceil = 10$.

Recursively defined function:

A function is said to be recursively defined if the function definition refers to itself. In order for the definition not to be circular, the function definition must have the following two properties:

1) There must be certain arguments, called *base values*, for which the function does not refer to itself.
2) Each time the function does refer to itself, the argument of the function must be closer to a base value.

A recursive function with these two properties is said to be *well-defined.*

Definition (Factorial Function):

a) If $n = 0$, then $n! = 1$.
b) If $n > 0$, $n! = n\,(n-1)!$

Definition (Fibonacci Function):

a) If $n = 0$ or $n = 1$, then $F_n = n$
b) If $n > 1$, then $F_n = F_{n-2} + F_{n-1}$.

Sequence:

A sequence is a function from the set of **N**= {1, 2, 3, ...} of positive integers to a set S. The notation a_n is used to denote the image of the integer n. Thus a sequence is usually denoted by $a_1, a_2, a_3, \ldots\ldots$ or $\{ a_n | n \in \mathbf{N}\}$ or simply $\{a_n\}$

- Sometimes the domain of a sequence is the set {0, 1, 2, ...} of nonnegative integers rather than **N**. In such a case we say n begins with 0 rather than 1.

Finite sequence:

A finite sequence over a set A is a function from {1, 2,... ...m} to A, and it is usually denoted by

$$a_1, a_2, \ldots \ldots, a_m.$$

- Such a finite sequence is sometimes called a *list* or an *m-tuple*.

Ex:

$a_n=1/n$	1, ½, 1/3,1/4, ...	(begins with n=1)
$b_n=2^{-n}$	1, ½, 1/4,1/8, ...	(begins with n=0)
$b_n=(-1)^{n+1}$	1, -1, 1,-1, ...	(begins with n=1)
$b_n=(-1)^n$	1, -1, 1,-1, ...	(begins with n=0)
$c_n=5^n$	1, 5, 25,125	(begins with n=0)

- Sequences of the form $a_1, a_2, \ldots \ldots, a_n$ are often used in computer science. These finite sequence are also called ***strings*** or ***word***. This string is also denoted by $a_1a_2 \ldots a_n$. The *length* of the string S is the number of terms in these string. The *empty string* is the string that has no terms. It has length zero.

Ex: The string **abcd** is a string of length 4.

Special integers sequences:

- See the Table-1 of page 72 of Discrete Mathematics & its Application by Kenneth H. Rosen.
- A wonderfully diverse collection of over 8000 different integer sequences has been constructed over the past 20 years by the mathematician Neil Sloane, who has teamed up with Simon Plouffe, to produce the *Encyclopedia of Integers Sequences*.

Summations:

Consider a sequence $a_1, a_2, a_3, \ldots$. Then the sums $a_1+ a_2+ a_3\ldots+ a_n$ and $a_m+a_{m+1}+\ldots\ldots+a_n$ will be denoted, respectively, by

$$\sum_{j=1}^{n} a_j \quad \text{and} \quad \sum_{j=m}^{n} a_j$$

Here j is called he dummy index or dummy variable or index of summation.

The uppercase Greek letter sigma, Σ, is used to denote summation.

Ex: $\sum_{j=1}^{5} j^2 = 1^2+2^2+3^2+4^2+5^2 = 55$

- $1+2+\dots\dots\dots+n = \frac{n(n+1)}{2}$
- $1^2+2^2+3^2+\dots\dots+n^2 = \frac{n(n+1)(2n+1)}{6}$
- $1^3+2^3+3^3+\dots\ +n^3 = \frac{n^2(n+1)^2}{4}$

Arithmetic Progression:

An Arithmetic Progression is a sequence of the form a, a+d, a+2d,………, a+nd where d is called the *common difference* of the progression.

- The nth term of A.P. is $T_n = a+(n-1)d$ and the common difference is $d = T_n - T_{n-1}$
- Sum of the first n terms of an Arithmetic Progression is $S_n = n/2\ \{2a+(n-1).d\}$

Geometric progression:

A Geometric Progression is a sequence of the form $a, ar, ar^2, \dots\dots, ar^k$ where a, the initial term, and r, the common ratio, are real numbers.

- The nth term of G.P. is $T_n = ar^{n-1}$ and the common ratio is $r = T_n / T_{n-1}$
- Sum of the first n+1 terms of an Geometric Progression is

$$S = \sum_{j=0}^{n} ar^j = \frac{ar^{n+1}-a}{r-1} \qquad \text{when } r \neq 1$$

- If r = 1, then S = (n + 1) a.
- $1+a+a^2+\dots\dots\dots+a^n = \frac{a^{n+1}-1}{a-1}$

Algorithm:

An algorithm M is a finite step-by-step list of well defined instructions for solving a particular problem.

Complexity of an Algorithm

The complexity of an algorithm M is the function $f(n)$ which gives the running time and/or storage space requirement of the algorithm in terms of the size n of the input data.

- The two cases one usually investigates in complexity theory are as follows:

  Worst case: The maximum value of $f(n)$ for any possible input.

  Average case: The expected value of $f(n)$.

Rate of Growth: Big O Notation:

Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that "$f(x)$ is of order $g(x)$" written as $f(x) = Og(x)$ [read as $f(x)$ is big oh of $g(x)$] if there exists a real number k and a positive constant C such that, for all x > k, we have

$$|f(x)| \leq C\,|g(x)|$$

Ex: $P(x) = O(x^m)$ where P(x) is m-degree polynomial e,g, $7x^2-9x+4 = O(x^2)$

- ***Rate of growth of some standard functions***: See page-65 of [SOS] Discrete mathematics by Lipschutz and Lipson.

(২)

2nd D.M

DM/MSA/L1

September 11, 1999

RUDIMENTS OF SET THEORY

DEFINITION OF A SET: A set is a collection of well-defined distinct objects. Constituents of a set are called objects, members, or elements.
The term object has been used without specifying what an object is.
This definition was first given by the German mathematician Georg Cantor in 1895.
Example: collection of chair, all real numbers etc.

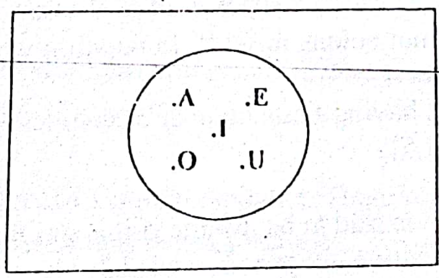
REPRESENTATION RULES OF SET:

1. Tabular or Roster method: The elements are represented by commas and enclosed between braces { }. Example: Set of all the districts in Bangladesh can be represented in this method by {Dhaka, Chittagong, Sylhet, Barisal, Rajshahi, Jessore}
2. Property or Rule method (Set builder notation): We characterize all those elements in the set by stating the property or properties they must have to become members. Example: The set D of all the districts in Bangladesh is: D = {x | x is a district of Bangladesh}

NOTATION: Sets are usually denoted by uppercase English letters such as A, B, X, Y etc.
The elements of sets are usually represented by lowercase letters such as a, b, x, y…
SOME STANDARD SETS:

a. $N \rightarrow$ The set of all natural numbers $\{\cancel{0},1,2,3\ldots\}$
b. $Z \rightarrow$ The set of all integers $\{0, \pm1, \pm2, \pm3\ldots\}$
c. $Z^{+} \rightarrow$ The set of all positive integers $\{1,2,3\ldots\}$
d. $R \rightarrow$ The set of all real numbers

VENN DIAGRAM METHOD: Named after English mathematician John Venn.
In Venn diagrams the *universal set* U, which contains all the objects under consideration is represented by a rectangle. Inside this rectangle, circles or other geometrical figures are used to represent sets. Sometime points are used to represent the particular element of the set.



U = set of 26 letters of English alphabet

Figure 1 : Venn diagram for the set of vowels

We write $a \in A$ to denote that a is an element of the set A. On the other hand, $a \notin A$ denotes that a is not an element of the set A.

SINGLETON SET: A set with exactly one element is called singleton set. E.g. B = {1}

NULL SET or **EMPTY SET** or **VOID SET:** A set which contains no elements is called null set. It is denoted by the Danish letter Ori ($\varnothing$) or by empty braces { }.

Example: A set of all positive integers that are greater than their square is a null set.

EQUALITY OF SETS: Two sets are equal if and only if they have the same elements.
Example: A = {1,3,5} and B = {3,1,5} are equal, i.e., A = B.
** The order in which the elements of a set are listed does not matter.
** It does not matter if an element of a set is listed more than once.
Example: A = {1,3,3,5,5,3} and B = {1,3,5} are equal.

SUBSET: The set A is said to be a subset of B if and only if every element of A is also an element of B.
We use the notation $A \subseteq B$ to indicate that A is a subset of the set B.
$A \subseteq B$ if and only if $\forall x\,(x \in A \rightarrow x \in B)$ is true.

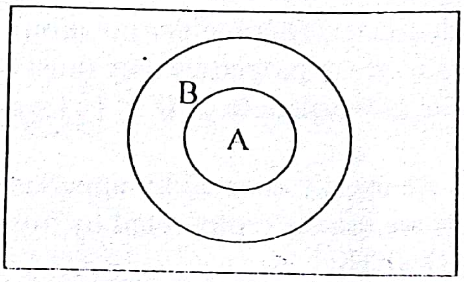


Figure 2 : Venn diagram showing that A is the subset of B

** The null set is a subset of every set: $\varnothing \subseteq S$
** Every set is the subset of itself.

PROPER SUBSET: If a set A is subset of B but $A \neq B$ then we say that A is a proper subset of B and denote this statement symbolically as $A \subset B$.
** If A and B are sets with $A \subseteq B$ and $B \subseteq A$ then A = B.

SUPERSET: A set C is called a superset of B if every elements of B belong to set C but every element of set C does not belong to set B. In notation we write this as $C \supset B$.

FINITE SET: The set having finite number of distinct elements is called Finite set.
Example: A = {1,2,3,4,5}

INFINITE SET: A set is said to be infinite if it is not finite.
Example: The set of positive integers $Z^+ = \{1,2,3\ldots\}$

CARDINALITY: Let S be a finite set of n elements. Then n is the cardinality of S and is denoted by $|S|$. If A = {1,2,3} then $|A| = 3$. In addition, for a null set we have $|\varnothing| = 0$.

POWER SET: IF A is a set, then the set of all subsets of A is called the power set of A and is denoted by P(A).
Example: Let A = {1,2,3}. Then P(A) = { ∅, {1},{2},{3},{1,2},{1,3},{2,3},{1,2,3} }.

Problem: Show that if $|A| = n$ then $|P(A)| = 2^n$.

SET OPERATIONS:

1. **UNION**: Let A and B be two sets. The union of the sets A and B, denoted by $A \cup B$ (also read as A cup B), is the set that contains those elements that are either in A or B, or in both.
   $A \cup B = \{x \mid x \in A \vee x \in B\}$
   Example: If A = {1,3,5} and B = {1,2,3,6} then $A \cup B$ = {1,2,3,5,6}

2. **INTERSECTION**: Let A and B be two sets. The intersections of the sets A and B denoted by $A \cap B$ is the set containing elements that are members of both A and B.
   $A \cap B = \{x \mid x \in A \wedge x \in B\}$
   Example: If A = {1,3,5} and B = {1,2,3,6} then $A \cap B$ = {1,3}

Problem: Show that $|A \cup B| = |A| + |B| - |A \cap B|$

DISJOINT SETS: Two sets A and B are disjoint if $A \cap B = \varnothing$.
Example: A = {1,3,5} and B = {2,4,6}. Since $A \cap B = \varnothing$, A and B are called disjoint.

DIFFERENCE: Let A and B be two sets. The difference of A and B, denoted by $A - B$, is the set containing those elements that are in A but not in B.
The difference of A and B is also called the *complement* of B *with respect to A*. (This is sort of a relative complement.)
$A - B = \{x \mid x \in A \wedge x \notin B\}$
** Note the phrasing "complement of B with respect to A". Later we will find a phrase "complement of B".

COMPLEMENT OF A SET: Let U be the universal set. The complement of the set A, denoted by $\overline{A}$ (or A' or A^c) is the difference of A with respect to U.
$\overline{A} = U - A = \{x \mid x \notin A\}$
Example: A = {a,e,i,o,u}, U = letters of English alphabet. Then $\overline{A}$ = {b, c, d, f, g, h, j…n, p…t, v…z}, i.e. $\overline{A}$ = {x | x is a English letter that is not a vowel}

ALGEBRAIC PROPERTIES OF SET OPERATION (SET IDENTITIES):

1. Identity Laws:
   $A \cup \varnothing = A$
   $A \cap U = A$
2. Domination Laws:
   $A \cup U = U$
   $A \cap \varnothing = \varnothing$

3. Idempotent Laws:

$A \cup A = A$

$A \cap A = A$

4. Complementation Law:

$\overline{\overline{A}} = A$

$A \cup \overline{A} = U$

$A \cap \overline{A} = \varnothing$

$\overline{\varnothing} = U$

$\overline{U} = \varnothing$

5. Commutative Laws:

$A \cup B = B \cup A$

$A \cap B = B \cap A$

6. Associative Laws:

$A \cup (B \cup C) = (A \cup B) \cup C$

$A \cap (B \cap C) = (A \cap B) \cap C$

7. Distributive Laws:

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

8. De Morgan's Theorem:

$\overline{A \cup B} = \overline{A} \cap \overline{B}$

$\overline{A \cap B} = \overline{A} \cup \overline{B}$

Problem: Prove De Morgan's theorems.

- Problem: Let A, B and C be three sets. Show that, $\overline{A \cup (\overline{B \cap C})} = (\overline{C} \cup \overline{B}) \cap \overline{A}$

SYMMETRIC DIFFERENCE: The ~ of A and B, denoted by $A \oplus B$ is the set containing those elements in either A or B, but not in both A and B.

COUNTING PRINCIPLE (For finite set):

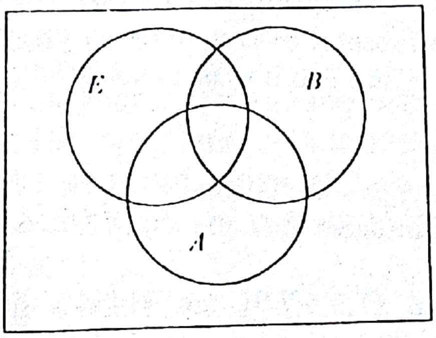
1. If A and B are disjoint finite set then $|A \cup B| = |A| + |B|$
2. If A and B are finite sets, then $|A \cup B| = |A| + |B| - |A \cap B|$
3. If A, B, and C are finite sets then
   $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$

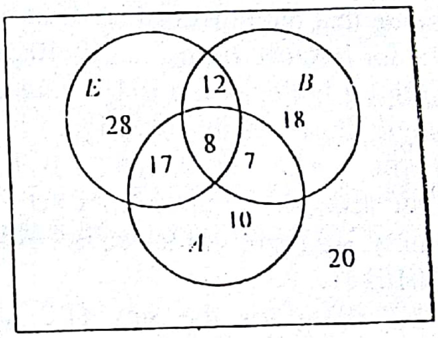
Problems: Draw Venn diagram for all of above operations.

Example: Consider the following data for 120 C.S.E. students at a university concerning the languages English, Bengali, and Arabic:

65 study English
45 study Bengali
42 study Arabic
20 study English and Bengali
25 study English and Arabic
15 study Bengali and Arabic
8 study all three languages.

Let E, B, and A denote the sets of students studying English, Bengali and Arabic, respectively. We wish to find the number of students who study at least one of the three languages, and to fill in the correct number of students in each of the eight regions of the Venn diagram shown in the fig. below.





We can find that $|E \cup B \cup A| = |E| + |B| + |A| - |E \cap B| - |E \cap A| - |B \cap A| + |E \cap B \cap A|$
$= 65 + 45 + 42 - 20 - 25 - 15 + 8 = 100$

That is, $|E \cup B \cup A| = 100$ students study at least one of the three languages.

We now use this to fill in the Venn diagram. We have:
8 study all three languages,
$20 - 8 = 12$ study English and Bengali but not Arabic.
$25 - 8 = 17$ study English and Arabic but not Bengali
$15 - 8 = 7$ study Bengali and Arabic but not English
$65 - 12 - 8 - 17 = 28$ study only English
$45 - 12 - 8 - 7 = 18$ study only Bengali
$42 - 17 - 8 - 7 = 10$ study only Arabic
$120 - 100 = 20$ do not study any of the languages.
Accordingly, the completed diagram appears in the fig. beside.
Observe that $28 + 18 + 10 = 56$ students study only one of the languages.

CARTESIAN PRODUCT: Let A and B be two sets. The Cartesian product of A and B, denoted by $A \times B$ is the set of all ordered pairs (a, b) *or* $\langle a, b \rangle$ where $a \in A$ and $b \in B$.
Hence, $A \times B = \{ (a, b) \mid a \in A \wedge b \in B \}$

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

- The Cartesian products $A \times B$ and $B \times A$ are not equal unless $A = \Phi$ ~~and~~ or $B = \Phi$ (so that $A \times B = \Phi$) or unless $A = B$.
- $|A \times B| = |A| . |B|$
- The Cartesian product $A \times A$ is also written as A^2, and similarly $A \times A \times A$ as A^3 and so on.

Problem: Find $A \times B \times C$ where $A = \{0,1\}$, $B = \{1,2\}$ and $C = \{a, b, c\}$.

COMPUTER REPRESENTATION OF SETS:

Assume that the universal set U is finite set. First, specify an arbitrary ordering of the elements of U, for instance $a_1, a_2, \ldots, a_n$. Represent a subset A of U with the bit string of length n, where the i-th bit in this string is 1 if a_i belongs to A and is 0 if a_i does not belong to A. The following example illustrates this technique.

Example: Let $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

The bit string that represent the set of odd integers in U, namely, {1, 3, 5, 7, 9} is 1010101010. Similarly, we represent the subset of all even integers in U, namely, {2, 4, 6, 8, 10}, by the string 0101010101.

The bit string for the sets {1,2,3,4,5} and {1,3,5,7,9} are 1111100000 and 1010101010, respectively.

The bit string for the union of these sets is, $1111100000 \vee 1010101010 = 1111101010$, which corresponds to the set {1, 2, 3, 4, 5, 7, 9}.

The bit string for the intersection of these sets is, $1111100000 \wedge 1010101010 = 1010100000$, which corresponds to the set {1, 3, 5}.

DM/MSA/L2
[redacted]

May [redacted]

INTRODUCTION TO LOGIC

[redacted]

WHAT IS LOGIC? Logic is the set of rules that enables us to determine whether any particular argument or reasoning is valid (correct). These rules are called **Rules of Inference**.

All of us are constantly using logical reasoning in our life (give some examples). **Logic is the mathematical model of the reasoning process, or careful argument**. Logic is concerned with all kinds of reasoning (it is a method of reasoning), whether they be legal arguments or mathematical proofs or conclusions in a scientific theory based upon a set of hypotheses.

[redacted]

We are constantly using logical reasoning without studying logic. Even other life forms can think logically (see the programs in National Geographic). However, logic has numerous applications in many different areas of human endeavor, which makes its study a very important one. Ideas of logic are central to the science of computing. Some examples of applications of logic are:

1. Logical reasoning is used

(i) In mathematics to prove theorems. If the compound proposition p is equivalent to TRUE (p is a tautology), then you have proved a theorem.

(ii) In the natural and physical sciences to draw conclusions from experiment.

(iii) In the social sciences and in our everyday lives to solve a multitude of problems.

2. Logic has numerous applications in computer science.

(i) The study of logic is vital in developing the means of communication with computing machines.
A machine can understand only when it is designed to understand. In order to design a machine that understands instructions we must first have a method of describing the meanings of instructions, and this is where logic comes into play. **<u>Logic is the set of rules that govern our understanding of meaning</u>**.

(ii) In computer design: A computer stores and computes with bits which are either 1 (true) of 0 (false), respectively by electrical signals. The computations are done with little pieces of hardware that, say, take two input bits and compute their "and" to get an output bit. Each little piece of hardware corresponds to one of the operations "and", "or", etc. Hence, computer hardware in effect evaluates compound propositions.
You will study this aspect of logic in details in CSE 2404 (Digital Logic Design).

(iii) Computer programs are constructed on the basis of logic. Looping and branching in a program are carried out using various Control Statements, which in turn are constructed through the rules of logic.

(iv) Correctness of computer programs is verified by analyzing them with the rules of logic.

PROPOSITION OR STATEMENT: A proposition is a declarative sentence that declares or proposes a single fact that is either true or false but not both.

Example:

1. Dhaka is the capital of Bangladesh
2. 1+1 = 3

Following are not propositions:

1. What time is it?
2. Read this carefully.
3. X + 1 = 2 (It is not a proposition unless X assigns a value).

These sentences do not declare any fact and consequently cannot be either true or false. Because these sentences are neither true nor false, they are not propositions.

Notation: In logic, we use lowercase letters such as p, q, r, ... to represent **propositional variables**, that is, variables that can be replaced by propositions. Thus we can write:

p: The sun is shining today.

q: It is hot.

Truth values: We say that a proposition has a truth value of true, denoted by **T** or **1**, if it is a true proposition. On the other hand, truth value of a proposition is false, denoted by **F** or **0**, if it is a false proposition.

LOGICAL OPERATORS AND CONNECTIVES: Logical operators are operators that are applied on existing proposition(s) to form new proposition(s). There are three basic types of logical operators: AND, OR and NOT. We will study their operation in details.

AND and OR operators connect two or more propositions and are also called **logical connectives.**

The operator NOT is different from other two logical operators. It does not join two propositions; rather it acts on a single proposition (unary operator) and modifies it. In this sense, it is not strictly correct to call it a connective. However, the use of the word connective is customary.

ATOMIC STATEMENT OR PRIMITIVE: A statement that does not contain any connectives is called an atomic or primary statement.

COMPOUND PROPOSITIONS: A compound proposition, also called a **molecular statement**, is a proposition formed from atomic statement(s) through the use of logical connectives and parentheses.

For example, if we connect p and q in the above example by connective AND, we get the compound statement **p AND q**: The sun is shining today and it is hot.

The truth value of a compound statement depends on:

(i) the truth values of the atomic statement (s) and

(ii) on the type(s) of connective(s) being used.

TRUTH TABLE: A truth table is a device that permits the truth value of a compound proposition to be found when the truth value(s) of the atomic statements(s) are given.

BASIC LOGICAL OPERATIONS: p AND q (conjunction), p OR q (disjunction), NOT p (negation); p XOR q (exclusive or).

CONJUNCTION: Let p and q be two propositions. The proposition "p and q" denoted by $p \wedge q$ is true when both p and q are true and is false otherwise.
Example: p: Today is Friday.
q : It is raining today.
$p \wedge q$: Today is Friday and it is raining today.
This is true on *rainy Fridays* and is false on any that is not a Friday or on Friday when it does not rain.

DISJUNCTION: Let p and q be two propositions. The proposition "p or q" denoted by $p \vee q$ is false when p and q are both false and is true otherwise.
Example: p: Today is Friday.
q : It is raining today.
$p \vee q$: Today is Friday or it is raining today.
It is true on any day that is either a Friday or a rainy day (including rainy Fridays).
It is false only on days that are not Fridays and it does not rain as well.

REMARK: The English word "or" is commonly used in two distinct ways.

First: In an Inclusive way. "p or q or both" i.e. at least one of the two alternatives occurs.
Example: "Students who have taken Programming -I or Programming - II can take this class." The students who have taken both courses can take the class, as well as the students who have taken just one of the two courses.

Second: In an Exclusive way. "p or q but not both", i.e. exactly one of the two alternatives can occur.
Example: "He will go to Dhaka or Sylhet." (Either Dhaka or Sylhet, but not both)
"Soup or salad comes with an entrée." (Customer can have either soup or salad, but not both)

EXCLUSIVE OR: Let p and q be two propositions. The exclusive or of p and q, denoted by $p \oplus q$ is the proposition that is true when exactly one of p and q is true and is false otherwise.

NEGATION: Let p be a proposition. The statement "it is not the case that p" is another proposition called negation of P. It is denoted by $\neg p$ and read as "not p".

Example: p: Today is Friday.
$\neg p$: It is not the case that today is Friday
Or $\neg p$ = Today is not Friday.

EXERCISE: Construct the truth tables of NOT, AND, OR and XOR operations.

FEW REMARKS ON LOGICAL OPERATORS:

1. In the literature, different symbols are used for negation, conjunction and disjunction (See Page-80 of SOS)
2. A given statement in the logic is denoted by a symbol, and it may correspond to several statements in English. This multiplicity happens because in a natural language one can express oneself in variety of ways.

Example: If p: I went to my class yesterday, then $\neg p$ can be any of the following:

i) It is not the case that I went to my class yesterday.

ii) I did not go to my class yesterday.

iii) I was absent from my class yesterday.

3. In the English language there is some ambiguity as to whether OR means ***either one or the other or both (inclusive or)*** or whether it means ***either one or the other but not both (exclusive or)***. The logical connective OR corresponds to the former whereas the logical connective XOR means the latter.
4. The symbol $\vee$ comes from the Latin word 'vel' which is the inclusive or.
5. The order of precedence of the logical connectives is NOT, AND, OR. This order of precedence helps us cut down the use of parentheses in a compound proposition.
6. There are 2^n possible combinations for n distinct components to obtain a truth table.

IMPLICATION or CONDITIONAL STATEMENT: Let p and q be two propositions. The implication $p \rightarrow q$ (read as p implies q) is the proposition that is false when p is true and q is false and true otherwise.

In this implication, p is called hypothesis (or antecedent or premise) and q is called the conclusion (or consequence).

Hence, an implication is a compound proposition where a conditional proposition implies a consequent proposition. Every implication can be posed in the form IF ... THEN ... An implication is also called a **conditional statement**.

Example: *"If I walk in the rain then I shall get wet."*

Sufficiency and Necessity: Within an implication $p \rightarrow q$, p is **sufficient** for the consequent proposition, that is if p is true then q will be also true. However, p is not **necessary** for q, i.e., q can be true even if p is false. So even if we know that q is true, we cannot say that p is true without knowing its truth value exactly.

There are many ways to express $p \rightarrow q$:

"if p then q"
"p implies q"
"if p, q"

"q if p"
"q whenever p"
"p is sufficient for q"

"p only if q"
"q is necessary for p"

REMARKS:

1. The mathematical concept of an implication is independent of a cause-and-effect relationship between hypothesis and conclusion. p and q need not have anything to do with one another in order to form the expression $p \rightarrow q$ (if p then q).

2. However, the if-then construction in many programming languages works in the customary sense and not in the sense of logic. The form of this construction is "**if** p **then** S" where p is a proposition and S is a program segment that is executed only when p is true.
3. The **converse** of $p \to q$ is $q \to p$; they need not be true at the same time. Ex: If I am exactly 18 years old then I am allowed to vote.
4. The **inverse** of $p \to q$ is $\neg p \to \neg q$. This is logically equivalent (defined later) to the converse.
5. The **contrapositive** of $p \to q$ is $\neg q \to \neg p$: these are same.
6. In fact, the implication connective is not a new one. It can be shown that $p \to q$ is same as $\neg p \vee q$.

BICONDITIONAL: Let p and q be two propositions. The biconditional $p \leftrightarrow q$ is the proposition that is true when p and q have the same values and is false otherwise.

It can be shown that this is same as $(p \to q) \wedge (q \to p)$. There are several ways to express it:

1. "p if and only if q" or "p iff q"
2. "p is necessary and sufficient for q"
3. "if p then q, and conversely".

Example: " T is a right triangle if and only if $a^2+b^2=c^2$".

Sufficiency and Necessity: Earlier we have seen in $p \to q$, p was sufficient but not necessary for q. However, q was necessary for p. If we apply this reasoning in the biconditional $p \leftrightarrow q$, we find that p is both necessary and sufficient for q. Also, q is necessary for p.

TAUTOLOGY: A compound proposition that is always true, no matter what the truth values of the propositions that occur in it is called a tautology.
Example: $p \vee \neg p$ (to be or not to be!) .

CONTRADICTON: A compound proposition that is always false is called a contradiction or absurdity.
Example: $p \wedge \neg p$.

CONTINGENCY: A proposition that is neither a tautology (i.e. not always true) nor a contradiction (i.e. not always false) is called a contingency. In other words a statement that can be either true or false, depending on the truth values of its propositional variables, is called a contingency.
Ex: $(p \vee q)$.

LOGICAL EQUIVALENCE: If two different compound propositions constructed from same atomic statements have identical truth table, then they have the same meaning. Two compound propositions p and q that have the same meaning are said to be logically equivalent, or simply equivalent or equal to each other. We denote that p is equivalent to q by $p \equiv q$ or by $p \Leftrightarrow q$.

From the definition of biconditional, it is clear that $p \leftrightarrow q$ is true whenever both p and q have the same truth values. Therefore, the propositions p and q are equivalent (i.e. have same truth values) if $p \leftrightarrow q$ is a tautology.
Example: $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$ (De Morgan's theorem)

Illustration of De Morgan's theorem:
Let p: Rahim goes to school and q: Karim goes to school. Then, $\neg(p \wedge q)$ = It is not the case that Rahim and Karim goes to school. $\neg p \vee \neg q$ = Rahim does not go to school or Karim does not go to school.

REMARKS:

1. Observe that $p \Leftrightarrow q$ is a statement in English (the metalanguage) and not in the logic (object language) (i.e. it is not a proposition). Also the symbol '$\Leftrightarrow$' is not a connective but a symbol in the metalanguage.
2. One way to determine whether two propositions are equivalent is to construct their truth table and compare them.
3. Another way, to show $p \Leftrightarrow q$ we need to show that $p \leftrightarrow q$ is a tautology. That is we need to show $(p \to q) \wedge (q \to p)$ is a tautology.

INTERLUDE: There are similarities among propositional calculus, set theory and Boolean algebra. In fact, all three have them rigorous mathematical connection. Following table summarizes their connection:

PROPOSITIONAL CALCULUS	SET THEORY	BOOLEAN ALGEBRA
~~CO~~NJUNCTION / OR ($\vee$) DIS $p \vee q$	UNION($\cup$) $P \cup Q$	OR (+) $p + q$
~~DI~~SJUNCTION, AND ($\wedge$) CON $p \wedge q$	INTERSECTION ($\cap$) $P \cap Q$	AND (·) p.q ; pq
NEGATION ($\neg$) $\neg p$	Complement $\overline{P}$	NOT $\overline{p}$

ALGEBRA OF PREPOSITIONS

Name	Equivalence	Comment
Identity laws	1. $P \vee F = p$ 2. $P \wedge T = p$	
Domination laws	1. $P \vee T = T$ 2. $P \wedge F = F$	
Idempotent laws	1. $p \vee p = p$ 2. $p \wedge p = p$	Idempotence of OR Idempotence of AND

Double negation law	$\neg(\neg p) = p$	
Commutative laws	1. $p \vee q = q \vee p$ 2. $p \wedge q = q \wedge p$	Commutativity of OR Commutativity of AND
Associative laws	1. $p \vee (q \vee r) = (p \vee q) \vee r$ 2. $p \wedge (q \wedge r) = (p \wedge q) \wedge r$	Associativity of OR Associativity of AND
Distributive laws	1. $p \vee (q \wedge r) = (p \vee q) \wedge (p \vee r)$ 2. $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$	Distributivity of OR over AND Distributivity of AND over OR
De Morgan's laws	1. $\neg (p \vee q) = \neg p \wedge \neg q$ 2. $\neg (p \wedge q) = \neg p \vee \neg q$	Complement of sum = Product of complements Complement of product = Sum of complements
Law of excluded middle	$p \vee \neg p = T$	These are two basic rules of classical logic and have been challenged by modern theories like fuzzy logic.
Principle of non-contradiction	$p \wedge \neg p = F$	
Absorption laws	1. $p \vee (p \wedge q) = p$ 2. $p \wedge (p \vee q) = p$	Problem 1.2.11 of Rosen.

Problem: Show that $\neg(p \vee (\neg p \wedge q))$ and $\neg p \wedge \neg q$ are logically equivalent.

TWO-STATE DEVICES AND PROPOSITIONAL LOGIC: The propositional logic is *two-valued* logic, because we admit only those statements having a truth value of true or false. An electrical switch has same property. The term **LOGIC GATE** is used to include a broader spectrum. It means a device which permits or stops the flow of not only electric current but any quantity that can go through the device, such as water, information, persons, etc. Chemical and physical processes are often controlled by such gates. Logic again plays a vital role in these situation.

The logical connectives AND and OR correspond to switches connected in series and in parallel.

A BIT OF HISTORY: Boolean Algebra was introduced by the English mathematician **George Boole** in 1854 in his book ***The Laws of Thought***. Boolean algebra is named in his honor.

Claude Elwood Shannon was the first person to apply Boolean algebra to the design of circuits. He was one of the first person to use bits to represent information. He started an extremely influential discipline called **Information Theory**. Modern computers could not have been developed without making use of the principles of Information Theory.

----------*----------

FURTHER LOGIC: PREDICATE CALCULUS

We are now going to study predicate logic or predicate calculus. Some of the motivations for studying this are:

1. So far we have considered propositions to which a truth value can be assigned immediately. Such propositions are very specific and, as a consequence, have a limited application. In reality, we need to be able to handle propositions whose truth value can only be ascertained when they have been tested against some external criterion. Example: x +3 >10. To cater for this situation we need to extend the propositional calculus to discuss propositions that are sometimes true and sometimes false.
2. We cannot point out the common features among propositions in the propositional calculus. Example: Rahim is a student. Karim is a student.
3. In many situations, propositional calculus does not enable us to make a conclusion from a given set of premises in a systematic way though we can do so intuitively. Predicate calculus is required for these cases.

   Example: All human being are mortal. John is a human being. Therefore, John is a mortal. Such a conclusion seems intuitively true. However, it does not follow from the inference theory of the statement calculus (we will study the inference theory of propositional and predicate calculus later).

PROPOSITIONS AND THEIR PREDICATES

In propositional calculus, our basic unit was atomic statement of which no analysis was admitted. In predicate calculus, we further divide an atomic statement into two parts: **subject** and **predicate**. Every proposition is a sentence containing a subject (or object) and a phrase that describes the subject. The descriptive phrase is called a predicate.
This separation of a proposition into subject and predicate is a most useful device that permits the creation of a more extensive logical object than a mere simple proposition.

TERMINOLOGIES and NOTATION:

(1) We symbolize a predicate by a uppercase English letter. Example: In the statement *Karim is a student* the predicate is "is a student" and can be symbolized by S.

(2) Every predicate describes something about one or more objects. Therefore, a statement could be written symbolically in terms of the predicate letter followed by the name or names of the objects to which the predicate is applied. If we denote Karim by k, then we have S(k). Hence, STATEMENT = PREDICATE (NAME)

(3) At this juncture, we introduce the notion of **propositional function** (we will write PF in short). If we insert variables in place of the names in a statement, we get a PF. Example: S(x) where x is some variable that can have as its value some name. Here x is used as a placeholder. Thus we can write: PROPOSITIONAL FUNCTION = PREDICATE (VARIABLES)

(4) Note that PF is different from proposition (or statement). There are two ways to convert a PF into a statement: (i) by value assignment and (ii) by using quantifiers.

VALUE ASSIGNMENT: A PF becomes a statement when the variables are replaced by the name of any object. Example: If we put x = Karim in the propositional function S(x), then we get a statement S (Karim).

The truth value of a statement obtained from a propositional function by substituting values (i.e.names) for the variables depends on these values, not on the predicate.
The truth or falsity of the proposition S(Karim) depends on Karim: he either is or is not a student.

Example: Let P(x) denote the statement " x>3 ". What are the truth values of P(4) and P(2) ?
Solution: The statement P(4) is obtained by setting x=4 in the statement " x > 3 ". Hence, P(4), which is the statement " 4 > 3 ", is true. However, P(2) , which is the statement " 2 > 3 " is false.

- A propositional function or predicate may have more than one variable. A PF which involve one variable is called one-place predicate and a predicate which involves n variables is called n-place predicate.

Example: Let Q(x,y) denote the statement "x = y + 3 ".What are the truth values of the propositions Q(1,2) and Q(3,0) ?
Solution: To obtain Q(1,2) , set x = 1 and y = 2 in the statement Q(x,y). Hence, Q(1,2) is the statement "1= 2 + 3 ", which is false. The statement Q(3,0) is the proposition "3 = 0+3", which is true.

PROGRAMMING CONSIDERATION: Propositional functions are commonly used in control structures in high-level programming languages. For example, a statement of the form

if x>3 then $y \leftarrow z$

includes the PF P(x) = 'x > 3'(Can you tell what is the predicate in this PF?). When the statement (i.e., an instruction in the program) is executed, the current value of x is inserted into P(x) and a statement is formed. The assignment statement $y \leftarrow z$ is executed if this statement is true.

QUANTIFIERS: Quantification is another way to create a proposition from a PF. There are two types of quantification, namely, Universal quantification & Existential quantification.

UNIVERSE OF DISCOURSE: The process of symbolizing statements in predicate calculus is sometimes made simpler by limiting the class of individuals or objects under consideration. This limitation means that the variables which are quantified stand only for those objects which are members of a particular set or class. Such a restricted class is called the universe of discourse.

Universal Quantification: The universal quantification of P(x) is the proposition

"P(x) is true for all values of x in the universe of discourse."

- The notation ∀x P(x) denotes the universal quantification of P(x). Here ∀ is called the *universal quantifier.*
- The proposition ∀x P(x) is also expressed as
  "for all x P(x)" or "for every x P(x)"

Example: Express the statement "Every student in this class has studied calculus" as a universal quantification.
Solution: Let P(x) denote the statement "x has studied calculus." Then the statement "Every student in this class has studied calculus" can be written as ∀x P(x), where the universe of discourse consists of the students in this class.

- This statement can also be expressed as ∀x (S(x) → P(x)) where S(x) is the statement "x is in the class." P(x) is as before , and the universe of discourse is the set of all students.

Example: Let P(x) be the statement "x + 1 > x". What is the truth value of the quantification ∀x P(x), where the universe of discourse is the set of real numbers ?
Solution: Since P(x) is true for all real numbers x, the quantification ∀x P(x) is true.

Example: Let Q(x) be the statement "x < 2". What is the truth value of the quantification ∀x Q(x), where the universe of discourse is the set of real numbers ?
Solution: Q(x) is not true for all real numbers x, since, for instance, Q(3) is false. Thus ∀x Q(x) is false.

- The universal quantification ∀x P(x) is the same as the conjunction
  $P(x_1) \wedge P(x_2) \wedge \ldots\ldots\ldots\ldots \wedge P(x_n)$

Example: What is the truth value of ∀x P(x) , where P(x) is the statement "$x^2 < 10$" and the universe of discourse consists of the positive integers not exceeding 4 ?
Solution: The statement ∀x P(x) is the same as the conjunction P(1) ∧ P(2) ∧ P(3) ∧ P(4), Since the universe of discourse consists of the integers 1, 2, 3, and 4. Since P(4), which is the statement "$4^2 < 10$", is false , it follows that ∀x P(x) is false.

Existential Quantification: The existential quantification of P(x) is the proposition
"There exists an element x in the universe of discourse such that P(x) is true."

- The notation ∃x P(x) denotes the existential quantification of P(x). Here ∃ is called the *existential quantifier.*
- The existential quantification ∃x P(x) is also expressed as
  "There is an x such that P(x)" or " There is at least one x such that P(x)" or " For some x P(x)."

Example: Let P(x) denote the statement "x > 3". What is the truth value of the quantification ∃x P(x), where the universe of discourse is the set of real numbers ?
Solution: Since "x > 3" is true, for instance, when x = 4 the existential quantification of P(x), which ∃x P(x), is true.

Example: Let Q(x) denote the statement "x = x + 1". What is the truth value of the quantification ∃x Q(x), where the universe of discourse is the set of real numbers ?
Solution: Since Q(x) is false for every real number x, the existential quantification of Q(x), which is ∃x Q(x), is false.

- The existential quantification ∃x P(x) is the same as the disjunction
  $P(x_1) \vee P(x_2) \vee \ldots\ldots\ldots\ldots \vee P(x_n)$

Example: What is the truth value of ∃x P(x) where P(x) is the statement "$x2 > 10$" and the universe of discourse consists of the positive integers not exceeding 4 ?
Solution: Since the universe of discourse is {1,2,3,4}, the proposition ∃x P(x) is the same as the disjunction P(1) ∨ P(2) ∨ P(3) ∨ P(4) ,
Since P(4), which is the statement "$4^2 > 10$", is true, it follows that ∃x P(x) is true.

Table-1: Page-25 [Rosen].

PROGRAMMING CONSIDERATION: It is sometimes helpful to think in terms of looping and searching when determining the truth value of a quantification. See page 25-26 [Rosen].

Translating Logical Expressions into Sentences

Example: Translate the statement $\forall x\, (C(x) \vee \exists y\, (C(y) \wedge F(x , y)))$ into English, where $C(x)$ is "x has a computer", $F(x , y)$ is "x and y are friends", and the universe of discourse for both x and y is the set of all students in your class.

Solution: The statement says that for every student x in your class x has a computer or there is a student y such that y has a computer and x and y are friends. In other words, every student in your class has a computer or has a friend who has a computer.

Example: Translate the statement $\exists x \forall y \forall z\, (((F(x , y) \wedge F(x , z) \wedge (y \neq z)) \rightarrow \neg F(y, z)))$ into English, where F(a,b) means a and b are friends and the universe of discourse for x, y, and z is the set of all students in your class.

Solution: This statement says that there is a student x such that for all students y and all students z other than y, if x and y are friends and x and z are friends, then y and z are not friends. In other words, there is a student none of whose friends are also friends with each other.

Translating Sentences into Logical Expressions

This type of translation is frequently used in mathematical statements, in logic programming, and in artificial intelligence.

Example: Express the statements "Some student in this class has visited Mexico" and "Every student in this class has visited either Canada or Mexico" using quantifiers.

Solution: Let the universe of discourse for the variable x be the set of students in your class. Let M(x) be the statement "x has visited Mexico" and C(x) the statement "x has visited Canada". The statement "Some student in this class has visited Mexico" can be written as $\exists x\, M(x)$.The statement "Every student in this class has visited either Canada or Mexico" can be written as $\forall x(C(x) \vee M(x))$.

Example: Express the statement "Everyone has exactly one best friend " as a logical expression.

Solution: Let B(x,y) be the statement "y is the best friend of x". To translate the sentence in the example, note that it says that for every person x there is another person y such that y is the best friend of x and that if z is a person other than y, then z is not the best friend of x. Consequently, we can translate the sentences as

$$\forall x\, \exists y \forall z\, (((B(x , y) \wedge ((z \neq y)) \rightarrow \neg B(x, z))).$$

- Implication is used in writing statements of the form "All A are B" while conjunction is used to symbolize statements of the form "Some A are B".

Remarks: Order of quantifiers in a multiply quantified propositional function is important, unless all the quantifiers are universal quantifiers or all are existential quantifiers.

Example: Let Q(x, y) denote "x + y = 0." What are the truth values of the quantification $\exists y \forall x\, Q(x, y)$ and $\forall x \exists y\, Q(x, y)$?

Solution: The quantification $\exists y \forall x\, Q(x, y)$ denotes the proposition

"There is a real number y such that for every real number x, Q(x, y) is true."

No matter what value of y is chosen, there is only one value of x for which x + y = 0. Since there is no real number y such that x+y = 0 for all real numbers x , the statement $\exists y \forall x\, Q(x, y)$ is false.

The quantification $\forall x \exists y\, Q(x, y)$ denotes the proposition

"For every real number x there is a real number y such that Q(x, y) is true."

Given a real number x, there is a real number y such that x + y = 0; namely, y = -x..

Hence, the statement $\forall x \exists y\, Q(x, y)$ is true.

- If $\exists y \forall x\ P(x, y)$ is true, then $\forall x \exists y\ P(x, y)$ must also be true. However, if $\forall x \exists y\ P(x, y)$ is true, it is not necessary for $\exists y \forall x\ P(x, y)$ to be true

Self Study: Example-24 Page-32 [Rosen].

Binding & Free Variables

When a quantifier is used on the variable x or when we assign a value to this variable, we say that this occurrence of the variable is *bound*. An occurrence of a variable that is not bound by a quantifier or set equal to a particular value is said to be *free*.
The *scope of a quantifier* is the statement immediately following the quantifier. If the scope is an atomic statement, then no parentheses are used to enclose the statement; otherwise parentheses are needed.

Problem: Determine which variables are free and which are bound in the following propositions.
(i) $\forall x\ P(x,y)$ (ii) $\forall x\ (P(x) \rightarrow Q(x))$ (iii) $\forall x\ (P(x) \rightarrow \exists y\ Q(x,y))$
(iv) $\exists x\ (P(x) \rightarrow Q(x))$ (v) $\exists x\ P(x) \rightarrow Q(x)$

Solution: In (i) x is bound variable and y is free variable. The scope of $\forall x$ is $P(x,y)$. Determine others by yourself.

Remarks: All the variables that occur in a PF must be bound to turn it into a proposition. This can be done using a combination of universal quantifiers, existential quantifiers, and value assignments.

Programming Consideration: For quantification of more than one variable, it is sometimes helpful to think in terms of nested loops. See Page-31, Table-2 [Rosen].

NEGATING QUANTIFIED PROPOSITIONS:

(a) Negating universally quantified propositions: When all of the elements in the universe of discourse can be listed (i.e., it is finite), then we can write $\forall x\ P(x) \Leftrightarrow P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)$ and $\exists x\ P(x) \Leftrightarrow P(x_1) \vee P(x_2) \vee \ldots \vee P(x_n)$ Hence,

$$\begin{aligned} \neg \forall x\ P(x) &\Leftrightarrow \neg (P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)) \\ &\Leftrightarrow \neg P(x_1) \vee \neg P(x_2) \vee \ldots \vee \neg P(x_n) \\ &\Leftrightarrow \exists x\ \neg P(x) \end{aligned}$$

Example: Consider the negation of statement " Every student in the class has taken a course in calculus". This statement is a universal quantification, namely $\forall x\ P(x)$ where $P(x)$ is the statement " x has taken a course in calculus". The negation of this statement is " It is not the case that every student in the class has taken a course in calculus". This is equivalent to " There is student in the class who has not taken a course in calculus". And this is simply the existential quantification of the negation of the original propositional function, namely, $\exists x\ \neg P(x)$.

(b) Negating existentially quantified propositions: Show that $\neg \exists x\ P(x) \Leftrightarrow \forall x\ \neg P(x)$.

Example: Consider the proposition "There is a student in this class who has taken a course in calculus." This is the existential quantification $\exists x\ P(x)$ where $P(x)$ is the statement " x has taken a course in calculus." The negation of this statement is the proposition "It is not the case that there is a student in this class who has taken a course in calculus." This is equivalent to " Every "student in this class has not taken calculus," which is just the universal quantification of the negation of the original propositional function, namely, $\forall x\ \neg P(x)$.

- Table-3 Page-33 {Rosen}
- If we call the existential and universal quantifiers dual of each other, then above two rules can be summarized by a mnemonic: *To Negate A Quantified Proposition, Replace The Quantifier With Its Dual And Negate The Propositional Function.*

Problem: Negate: Every course has a component that is an elective. Hint: $\neg(\forall x\ \exists y\ P(x,y))$.

RELATION

The concept of a relation is a basic concept in everyday life as well as in mathematics. We have already used various relations. Associated with a relation is the act of comparing objects, which are related to one another. The ability of a computer to perform different tasks based upon the result of a comparison is one of its most important attributes, which is used several times during the execution of a typical program.

The word "relation" suggests some familiar examples of relations such as the relation of father to son, mother to son, brother to sister, etc. Familiar examples in arithmetic are relations such as "greater that," "less than, " or that of equality between two real numbers. We also know the relation between the area of a circle and its radius and between the area of a square and its side. These examples suggest relationship between two objects. The relation between parents and child, the coincidence of three lines, and that of a point lying between two points are examples of relations among three objects. Similar examples exist for relations among four or more objects.

Definition:

Let A and B sets. A *binary relation* or simply *relation* from A to B is a subset of A × B.

Suppose R is a relation from A to B. Then R is a set of ordered pairs where each first element comes from A and each second element comes from B.

- We use the notation ${}_aR_b$ (read as "a is R-related to b") to denotes that $(a,b) \in R$. and ${}_a\not R_b$ (read as "a is not R-related to b") to denotes that $(a,b) \notin R$.

Domain & Range of a relation:

The *domain* of relation R is the set of all first elements of the ordered pairs, which belongs to R. The *range* of relation R is the set of all second elements of the ordered pairs, which belongs to R.

Example: Let $A = \{1,2,3\}$ and $B = \{x,y,z\}$ and let $R = \{(1,y), (1,z), (3,y)\}$. Then R is a relation from A to B since R is a subset of A × B. With respect to this relation,

$1Ry,\ 1Rz,\ 3Ry$ but $1\not Rx,\ 2\not Rx,\ 2\not Ry,\ 2\not Rz,\ 3\not Rx,\ 3\not Rz$

The domain of the relation R is $\{1,3\}$ and the range is $\{y,z\}$.

Example: Let A={eggs, milk,corn} and B={cows, goats, hens}. We can define a relation R from A to b since by $(a.b) \in R$ if a is produced by b. In other words,

R = {(eggs, hens), (milk, cows), (milk, goats)}

Inverse Relation:

Let R be any relation from set A to Set B. The inverse of R, denoted by R^{-1}, is the relation from B to A which consists of those ordered pairs which, when reversed, belong to R; that is

$$R^{-1} = \{(b,a) \mid (a,b) \in R\}$$

Example: The inverse of the relation $R = \{(1,y), (1,z), (3,y)\}$ from $A = \{1,2,3\}$ to $B = \{x,y,z\}$ follows

$$R^{-1} = \{(y,1), (z,1), (y,3)\}$$

- The domain and range of R^{-1} are equal to the range and domain of R respectively.
- If R is any relation, then $(R^{-1})^{-1} = R$.
- If R is a relation on A, then R^{-1} is also a relation on A.

Relation on a Set:
A relation on the set A is a relation from A to A. In other words, a relation on the set A is a subset of A × A.

Example: Let A be the set {1,2,3,4}. The relation R = {(a, b) | a divides b} will be
R = {(1,1), (1,2), (1,3), (1,4), (2,2), (2,4), (3,3), (4,4)}

- Let A be any set. Then A × A and ∅ are subsets of A × A and hence are relations on A called the *universal relation* and *empty relation*, respectively.
- Let A be any set. The relation R = {(a,a) | a ∈ A} on A is called the *identity* or *diagonal relation* on A and it will be denoted by Δ_A or simply Δ.

Representations of Relations on Finite Sets

There are many ways to represent a relation between finite sets. Some of the methods are discussed bellow.

a. Representing Relations using Matrices:

A relation between finite sets can be represented using a zero-one matrix. Suppose that R is a relation from A = {a_1, a_2, …., a_m} to B = {b_1, b_2, ….,b_n}. (Here the elements of the sets A and B have been listed in a particular, but arbitrary, order. Furthermore when A = B we use the same ordering for A and B.) The relation R can be represented by the matrix $M_R = [m_{ij}]$,

where $m_{ij} = 1$ if $(a_i,b_j) \in R$
$\qquad\qquad 0$ if $(a_i,b_j) \notin R$

Example: Let A = {1,2,3} and B = {x,y,z} and let R = {(1,y), (1,z), (3,y)}. Then the Matrix representation of this relation R is

	x	y	z
1	0	1	1
2	0	0	0
3	0	1	0

Example: Let A = {a_1, a_2, a_3} and B = {b_1, b_2, b_3, b_4, b_5}. Which ordered pairs are in the relation R represented by the matrix M_R =

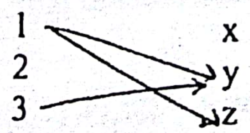
	b_1	b_2	b_3	b_4	b_5
a_1	0	1	0	0	0
a_2	1	0	1	1	0
a_3	1	0	1	0	1

Solution: Since R consists of the ordered pairs (ai, bj) with mij = 1, it follows that
R = {(a_1, b_2), (a_2, b_1), (a_2, b_3), (a_2, b_4), (a_3, b_1), (a_3, b_3), (a_3, b_5)}

b. Representing Relations using Arrow diagram:

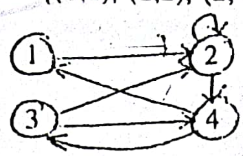
In this representation the elements of A and the elements of B are write down in two disjoint disks, and then draw an arrow from a ∈ A to b ∈ B whenever a is related to b.

Example: Let A = {1,2,3} and B = {x,y,z} and let R = {(1,y), (1,z), (3,y)}. Then the arrow diagram of this relation R is



c. Representing Relations using Directed graphs:
There is another way of representing a relation R when R is a relation from a finite set to itself. In this representation, each element of the set is represented by a point, and each ordered pair is represented using an arc with its direction indicated by an arrow. This diagram is called the directed graph of the relation.

Example: Let A = {1,2,3,4} and R = {(1,2), (2,2), (2,4), (3,2), (3,4), (4,1), (4,3)}. Then the directed graph of this relation R is



- A *directed graph* or *digraph* consists of a set V of *vertices* (or nodes) together with a set E of ordered pairs of elements of V called *edges* (or arcs). The vertex a is called the *initial vertex* of the edge (a,b) and the vertex b is called the *terminal vertex* of this edge.
- An edge of the form (a,a) is represented using an arc from the vertex a back to itself. Such an edge is called a *loop*.

Example 8, 9 : Page - 394 (Rosen).

Functions as Relations:
We know, A function *f* from a set A to a set B assigns a unique element of B to each element of A. The graph of *f* is the set of ordered pairs (a,b) such that b = *f* (a). Since the graph of *f* is a subset of A × B, it is a relation from A to B. Moreover, the graph of a function has the property that every element of A is the first element of exactly one ordered pair of the graph. Conversely, if R is a relation from A to B such that every element in A is the first element of exactly one ordered pair of R, then a function can be defined with R as its graph. This can be done by assigning to an element a of A the unique element a of A the unique element b ∈ B such that (a,b) ∈ R.

Example: Consider the following relation on the set A = {1, 2, 3:}

f = {(1, 3), (2, 3), (3, 1)}
g = {(1, 2), (3, 1)}
h = {(1, 3), (2, 1), (1, 2), (3, 1)}

f is a function from A into A since each member of A appears as the first coordinate in exactly one ordered pair in *f*. *g* is not a function from A into A since 2 ∈ a is not the first coordinate of any pair in *g* and so *g* does not assign any image to 2. Also *h* is not a function from A into A since 1 ∈A appears as the first coordinate of two distinct ordered pairs in *h*, (1,3) and (1,2). If *h* is to be a function it cannot assign both 3 and 2 to the element 1 ∈ A.

Problem: How many relations are there on a set with n elements?
Solution: A relation on a set A is a subset of A × A. Since A × A has n^2 elements when A has n elements, and A set with m elements has 2^m subsets, hence there are 2^{n^2} subsets of A × A. Thus, there are 2^{n^2} relations on a set with n elements.

Properties of Relations

There are several properties that are used to classify relations on a set. We will introduce the most important of these here.

Reflexive Relations:

A relation R on a set A is called *reflexive* if (a,a) ∈ R for every elements a ∈ A. Thus R is not reflexive if there exists an a ∈ A such that (a,a) ∉ R.

Example 2.6: Consider the following five relations on the set A = {1,2,3,4}:

R_1 = {(1,1), (1,2), (2,3), (1,3), (4,4)} as, t.
R_2 = {(1,1), (1,2), (2,1), (2,2), (3,3),(4,4)} R, S, t
R_3 = {(1,3), (2,1)} as,
R_4 = ∅ [Empty Relation] R, S, as.
R_5 = A × A [Universal Relation] S, R, t

Determine which of the relations are reflexive.

Solution: Here R_2 and R_5 are reflexive.

(1,1)(2,2)

Example: Is the "divides" relation on the set of positive integers reflexive?
Solution: Since a|a whenever a is positive integer, the "divides" relation is reflexive.

Example 2.7: Page-32 (SOS)

(A×A) — ∅ (2,2)
(1,2) (2,1)

Symmetric and Antisymmetric Relations:

A relation R on a set A is called *symmetric* if (b, a) ∈ R whenever (a, b) ∈ R, for a,b ∈ R. A relation R on a set A such that (a, b) ∈ R and (b, a) ∈ R only if a = b, for a, b ∈ R, is called *antisymmetric*.

(1,2) (2,1)

Example: R_2, R_4 and R_5 of example 2.6 are symmetric but R_1 and R_3 is not symmetric.

Example: R_1, R_3 and R_4 of example 2.6 are antisymmetric but R_2 and R_5 is not antisymmetric.

Example: Is the "divides" relation on the set of positive integers symmetric? Is it antisymmetric?

Solutions: This relation is not symmetric since 1|2 but not 2 | 1.
It is antisymmetric, for if a and b are positive integers with a|b and b|a, then it must be a = b.

- The term symmetric and antisymmetric is not opposite. For example, R = {(1, 3), (3, 1), (2, 3)} is neither symmetric nor antisymmetric. On the other hand, R = {(1, 1), (2, 2)} is both symmetric and antisymmetric.

(a,b) (b,c) (c,a)
{1,2} {2,3} {3,1}

Transitive Relations:

A relation R on a set A is called transitive if whenever (a, b) ∈ R and (b, c) ∈ R then (a, c) ∈ R, for a, b, c ∈ A.

Example: R_1, R_2, R_4 and R_5 of example 2.6 are transitive but R_3 is not transitive.

Example: Is the "divides" relation on the set of positive integers transitive?

Solution: Suppose that a divides b and b divides c. Then there are positive integers k and l such that b = (a * k) and c = (b * l). Hence, c = (a * k * l), so that a divides c. It follows that this relation is transitive.

(1,1), (1,2) (1,2)
a = b

(a,b) ∈ R
(b,a) ∈ R
a = b

- *Example: How many reflexive relations are there on a set with n elements?*
Solutions: $2^{n(n-1)}$. [see Page - 380 (Rosen)]

Combining Relations

Since relations from A to B are subsets of A × B, two relations can be combined in any way two sets can be combined. Consider the following example.

Example: Let A = {1, 2, 3} and B = {1, 2, 3, 4}. The relations R_1 = {(1, 1), (2, 2), (3, 3)} and R_2 = {(1, 1), (1, 2), (1, 3), (1, 4)} can be combined to obtain

- $R_1 \cup R_2$ = {(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (3, 3)}
- $R_1 \cap R_2$ = {(1, 1)}
- $R_1 - R_2$ = {(2, 2), (3, 3)}
- $R_2 - R_1$ = {(1, 2), (1, 3), (1, 4)}
- $R_1 \oplus R_2$ = {(1, 2), (1, 3), (1, 4), (2, 2), (3, 3)}

Composition of Relation

There is another way that relations are combined which is analogous to the composition of function.

Definition: Let R be a relation from a set A to a set B and S is a relation from B set to a set C. The *composite* (also called composition) of R and S is the relation consisting of ordered pairs (a, c), where a ∈ A, c ∈ C, and for which there exists an element b ∈ B such that (a, b) ∈ R and (b, c) ∈ S. We denote the composite of R and S by S o R.

S o R = {(a, c) | b ∈ B for which (a, b) ∈ R and (b, c) ∈ S}

□ Many texts denote the composition of relations R and S by R o S.

Example: Let A = {1, 2, 3, 4}, B = {a, b, c, d} and C = {x, y, z} and let R = {(1, a), (2, d), (3, a), (3, b), (3, d)} and S = {(b, x), (b, z), (c, y), (d, z)}. Find S o R.
Solution: S o R = {(2,z), (3,x), (3,z)} [page - 31 (SOS)]

Powers of Relation:

The powers of relation R can be inductively defined from the definition of composite of two relations.

Definition: Let R be a relation on the set A. The powers R^n, n = 1,2,3,...., are defined inductively by $R^1 = R$ and $R^{n+1} = R^n \circ R$.

□ The definition shows that $R^2 = R \circ R$, $R^3 = R^2 \circ R = (R \circ R) \circ R$.

Example: Let R = {(1, 1), (2, 1), (3, 2), (4, 3), ~~(2, 2)~~}. Find the powers R^n, n = 2,3,4,......
Solution: Since $R^2 = R \circ R$, we find that R^2 = {(1,1), (2,1), (3,1), (4,2)}. Furthermore, since $R^3 = R^2 \circ R$, R^3 = {(1,1), (2,1), (3,1), (4,1)}. Additional computation shows that R4 is the same as R3, so R^4 = {(1,1), (2,1), (3,1), (4,1)}. It also shows that $R^n = R^3$ for n = 5,6,7,...

Theorem: The relation on a set A is transitive if and only if $R^n \subseteq R$ for n = 1,2,3,...
Proof: Page - 382 (Rosen).

n-ARY Relations:

Let A_1, A_2, ..., A_n be sets. An n-ary relation on these sets is a subset of $A_1 \times A_2 \times \ldots \times A_n$. These sets A_1, A_2, ..., A_n are called the *domains* of the relation and n is called its degree.

Example: Let R be the relation consisting of triples (a,b,c) where a, b and c are integers with a < b < c. Then (1,2,3) ∈ R, (but 2,4,3) ∉ R. The degree of relation is 3. Its domains are equal to the set of integers.

Self study: Example 3, 4, 5, 6 [Page -391-93 (Rosen)].

Equivalence Relations

A relation R on a set A is called an *equivalence relation* if it is reflexive, symmetric and transitive.

Example: Suppose that R is the relation on the set of strings of English letters such that aRb if and only if l(a) = l(b), where l(x) is the length of the string x. Is R an equivalence relation?

Solution: Since l(a) = l(b), it follows that a R a whenever a is string, so that R is reflexive. Next, suppose that a R b, so that l(a) = l(b). Then b R a, since l(b) = l(a). Hence, R is symmetric. Finally, suppose that a R b and b R c. Then l(a) = l(b) and l(b) = l(c). Hence, l(a) = l(c), so that a R c. Consequently, R is transitive. Since R is reflexive, symmetric and transitive, it is an equivalence relation.

Example 2.13: Page 36 (SOS).

Partition

Let S be a non-empty set. A partition of S is a subdivision of S into non-overlapping, nonempty subset. Precisely, a *partition* of S is a collection $\{A_i\}$ of non-empty subsets of S such that

i) Each a in S belongs to one of the A_i

ii) The sets of $\{A_i\}$ are mutually disjoint i.e. if $A_i \neq A_j$ then $A_i \cap A_j = \varnothing$.

In other words, a partition P of S is a subdivision of S into disjoint non-empty sets.

Example: Consider the following collections of subsets S = {1, 2, 3,………,8, 9}:

i) [{1,3,5}, {2,6}, {4,8,9}]

ii) [{1,3,5}, {2,4,6,8}, {5,7,9}]

iii) [{1,3,5}, {2,4,6,8}, {7,9}]

Then (i) is not a partition of S since 7 in S does not belongs to any of the subsets. Furthermore, (ii) is not a partition of S since {1,3,5} and {5,7,9} are not disjoint. On the other hand, (iii) is a partition of S,

Exercise: Find all partitions of S = {1,2,3}

Classes of Sets:

Let S be a set. Then *class of sets* is the collection of some subsets of S. If we consider some of the sets in a given class of sets, then it is called *subclass of sets*.

Example: Suppose S = {1,2,3,4}. Let A be the class of subsets of S which contain exactly three elements of S. Then A = [{1,2,3}, {1,2,4}, {1,3,4}, {2,3,4}]. Let B be the class of subsets of S which contain 2 and two other elements of S. Then B = [{1,2,3}, {1,2,4}, {2,3,4}]. Here, B is the subclass of A, since every element of B is also an element of A.

Equivalence Classes

Let R be an equivalence relation on set A. The Set of all elements that are related to an element *a* of A is called the *equivalence class* of *a*. The equivalence class of *a* with respect to R is denoted by $[a]_R$. When only one relation is under consideration we will skip the subscript operator R and write just [a] for this equivalence class.

In other words, if R is an equivalence relation on a Set A, the equivalence class of the element a is $[a] = \{s \mid (a,s) \in R\}$

If $b \in [a]_R$, b is called a *representative* of this equivalence class.