# Scilab Textbook Companion for Discrete Mathematics And Its Applications by K. H. Rosen[1]

Created by
Sai L
BTech
Electronics Engineering
SNIST
College Teacher
None
Cross-Checked by
None

July 31, 2019

[1] Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

# Book Description

**Title:** Discrete Mathematics And Its Applications

**Author:** K. H. Rosen

**Publisher:** McGraw Hill Higher Education

**Edition:** 7

**Year:** 2012

**ISBN:** 9780071315012

Scilab numbering policy used in this document and the relation to the above book.

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

**AP** Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

# Contents

# List of Scilab Codes

# Chapter 1

# The Foundations Logic and Proofs

**Scilab code Exa 1.1** `propositions`

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;
s1=1+1==2
s2=2+2==3
mprintf("The following sentences are Propositions\n") //Proposition should be a declarative sentence or should result in either a YES or a NO.
mprintf("1.  Washington D.C is the capital of the United States of America\n")
mprintf("2.  Toronto is the capital of Canada\n")
mprintf("3.  1+1=2 %s ", string([%T]))
mprintf("\n4.  2+2=3 %s ", string([%F]))
//Since these statements are declarative and they answer the question YES or NO they are called propositions.
```

**Scilab code Exa 1.2** `propositions`

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

mprintf("1.  What time is it? \n")
mprintf("2.  Read this carefully. \n")
mprintf("3.  x+1=2.\n")
mprintf("4.  x+y=Z.\n")
mprintf("Sentences 1 and 2 are not propositions since they are not declarative.\nSentences 3 and 4 are neither true nor false and so they are not propositions.")
```

---

**Scilab code Exa 1.3** `Negation`

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

mprintf("Propositon p=Michael s PC runs Linux.")
mprintf("\n Negation of p is ~p : It is not the case that Michael s PC runs Linux.")
mprintf("\n Negation of p is ~p : Michael s PC does not run.Linux")//Negation is opposite of the truth value of the proposition expressed with "it is not the case that" or with "not".
```

---

**Scilab code Exa 1.4** Negation

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

mprintf( "Let p=Vandana s smartphone has at least 32 GB of memory.")
mprintf( "\nThe negation of p is ( ~p ) :It is not the case that Vandana s smartphone has at least 32GB of memory.")
mprintf( "\nOr in simple English ( ~p ): Vandana s smartphone does not have at least 32GB of memory.")
mprintf( "\nOr even more simple as ( ~p ): Vandana s smartphone has less than 32GB of memory.")
```

**Scilab code Exa 1.5** Conjunction

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

p="Rebecca s PC has more than 16GB free hard disk space"
q="The processor in Rebecca s PC runs faster than 1 GHz"
mprintf("Let p,q be two propositions")
mprintf("\nLet p=%s \n Let q=%s",p,q)
mprintf("\nConjunction of p^q is : %s and %s",p,q) //conjunction combines two propositons with "and"
```

**Scilab code Exa 1.6** Disjunction

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

p="Rebecca s PC has more than 16GB free hard disk space"
q="The processor in Rebecca s PC runs faster than 1 GHz"
mprintf("Let p,q be two propositions")
mprintf("\nLet p= %s\n Let q=%s",p,q)
mprintf("\nDisjunction of pVq is : %s or %s",p,q) // cup symbol.= V
//Disjunction combines two propositons using OR
```

**Scilab code Exa 1.7** conditional statements

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

p="Maria learns discrete mathematics"
q="Maria will find a good job"
mprintf("Let p=%s \n Let q=%s",p,q)
mprintf("\np->q is : If %s then %s",p,q) //p->q p implies q means If p then q.
mprintf("\np->q is also expressed as :%s when %s",q,p)
```

**Scilab code Exa 1.8** bitwise operations

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

x = [0 1 1 0 1 1 0 1 1 0];
y = [1 1 0 0 0 1 1 1 0 1];

bit_and=bitand(x,y)
bit_or=bitor(x,y)
bit_xor=bitxor(x,y)

disp(bit_and,"The bitwise AND is")
disp(bit_or,"The bitwise OR is")
disp(bit_xor,"The bitwise XOR is")
```

**Scilab code Exa 1.9** check whether the given statements are true

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

function p(x) //function definition to check whether the given statements are true.
if(x>3) then
    mprintf("\np(%d) which is the statement %d > 3, is true",x,x)
```

```
else
    mprintf("\np(%d) which is the statement %d > 3, is false",x,x)
end
endfunction

p(4)
p(2)
```

**Scilab code Exa 1.10** check whether systems are under attack

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

function atck(X)
    if (X=='CS1') then
        mprintf("\nA(%s) is true",X)
    elseif (X=='MATH1') then
        mprintf("\nA(%s) is true",X)
    else
        mprintf("\nA(%s) is false",X)
        end
endfunction

//Defining systems to check whether they are under attack through a function.
x1='CS1'
x2='CS2'
x3='MATH1'

atck(x1)
atck(x2)
atck(x3)
```

```

mprintf("\nSystems under attack are CS1 and MATH1.\nThe truth values for the same are calculated using functions.")
```

**Scilab code Exa 1.11** sqrt proposition

```
//Chapter 01: The Foundations: Logic and Proofs

clc;
clear;

v1=sqrt(2)
v2=(3/2)

//let p be the proposition that sqrt(2) > (3/2)
if v1 > v2 then              //which is false
    z=v1**2 >v2**2
    mprintf("(sqrt(2))^2 > (3/2)^2 %s ", string([%F]))//which is false and as a result will not be printed
end

//The conclusion is false,therefore final argument
fin_arg=v1**2>v2**2//sqrt(2)^2 is less than (3/2)^2
disp(fin_arg)
```

# Chapter 2

# Basic Structures Sets Functions Sequences Sums and Matrices

**Scilab code Exa 2.1** factorial

```
//Chapter 02:Basic Structures: Sets, Functions,
   Sequences, Sums and Matrices

clc;
clear;

mprintf("The factorial of 1 is")
disp(factorial(1))
mprintf("The factorial of 2 is")
disp(factorial(2))
mprintf("The factorial of 6 is")
disp(factorial(6))
mprintf("The factorial of 20 is")
disp(factorial(20))

disp("It shows that the factorial function grows
   extremely rapidly as the number grows.")
```

**Scilab code Exa 2.2** generate a sequence

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

//To generate a sequence a_n=1/n
i=1.0 //floating point division
n=input("Enter the number of terms in the sequence:");
mprintf("\na_n=1/n")
mprintf("\nWhen n=%d   a_n is:",n)
for i=1:n //iteration till the number of terms specified by the user
a=1.0/i
mprintf( "\n1/%d,\t",i)
end
for i=1:n //iteration till the number of terms specified by the user
a=1.0/i
mprintf("\n%f,\t",a)
end
```

**Scilab code Exa 2.3** generate the GP

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;
```

```

n=input("Enter the no. of terms in the sequence to generate the GP:");
i=1
mprintf("\nThe list of terms:")
for i=0:n
    mprintf("b%d ,\t",i)
end
mprintf("begins with  ")
for i=0:n //iterate for the number of terms given as input
    b_n=(-1)**i
    mprintf("%d ,",b_n)
end
mprintf("\nThe list of terms:")
for i=0:n
    mprintf("c%d ,\t",i)
end
mprintf("begins with  ")
for i=0:n  //iterate for the number of terms given as input
    c_n=2*(5**i)
    mprintf("%d ,",c_n)
end
mprintf("\nThe list of terms:")
for i=0:n
    mprintf("c%d ,\t",i)
end
mprintf("begins with  ")
for i=0:n  //iterate for the number of terms given as input
    d_n=6.0*((1.0/3.0)**i)
    mprintf("%f ,",d_n)  //prints the fraction values in decimals.  Floating point division
end
```

**Scilab code Exa 2.4** generates the sequence

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

n=input("Enter the number terms in the sequence:");
s_n=-1+4*n
t_n=7-3*n
i=0
mprintf("The list of terms:")
for i=0:n-1
    mprintf("s%d ,",i)
end
mprintf("   begins with   ")
for i=0:n-1           //generates the sequence for -1*4i
    t=-1+4*i
    mprintf("%d ,",t)
end
mprintf("\nThe list of terms:")
for i=0:n-1
    mprintf("t%d ,",i)
end
mprintf("   begins with   ")
for i=0:n-1            //generates the sequence for 7-3i
    t=7-3*i
    mprintf("%d ,",t)
end
```

**Scilab code Exa 2.5** `Length of the string`

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

str=['abcd']
disp(length(str),'Length of the string is:')
```

**Scilab code Exa 2.6** `display list`

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

a=[2,0,0,0] //given
//index starts from 1 so a0 is not present
for i=2:4
    a(i)=a(i-1)+3
    mprintf("a[%d]=%d\n",i,a(i))
end

mprintf("\nOriginal List:\n")
for i=1:4
mprintf("a[%d]=%d\n",i,a(i))
end
```

**Scilab code Exa 2.7** `display list`

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

a=[3,5,0,0] //given
//index starts from 1 so a0 is not present
for i=3:4
    a(i)=a(i-1)-a(i-2)
    mprintf(”a[%d]=%d\n”,i,a(i))
end
```

**Scilab code Exa 2.8** `Fibonacci series`

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

f=[0,1,0,0,0,0,0] //given
//index starts from 1 so f0 is not present
mprintf(”Fibonacci series is:\n”)
for i=3:7
    f(i)=f(i-1)+f(i-2)
    mprintf(”f[%d]=f[%d] + f[%d]=%d\n”,i,i-1,i-2,f(i))
end
```

**Scilab code Exa 2.9** factorial

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

n=1
result=0
number=input("Enter the number:");
for i=1:number-1
n=n+(i*n)
end
mprintf("The factorial of %d is %d",number,n)
```

**Scilab code Exa 2.10** print list

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

a=[]
i=1
for i=1:10
    for j =1:i
        mprintf("%d   ",i)
end
end
```

**Scilab code Exa 2.11** summation of j power 2

```
//Chapter 02:Basic Structures: Sets, Functions,
   Sequences, Sums and Matrices

clc;
clear;

//Finding the summation of j**2
up=input("Enter the upper limit for the operation j
   **2:");
low=input("Enter the lower limit for the operation j
   **2:");
sum_j=0
mprintf("\nThe square of terms from 1 to n :\n")
for j=low:up
mprintf("%d **2 +",j),
    j=j**2
    sum_j=sum_j+j
end
mprintf("=%d",sum_j)
```

**Scilab code Exa 2.12** value for the sequence

```
//Chapter 02:Basic Structures: Sets, Functions,
   Sequences, Sums and Matrices

clc;
clear;

k=4 //lower limit
sum_a=0
```

```
mprintf("The value for the sequence ")
for k=4:8
    if (k==8) then
        mprintf("(-1) ** %d ",k)
        else
    mprintf("(-1) ** %d +",k)
end
sum_a=sum_a + ((-1) ** k)
end
mprintf("=%d",sum_a)
```

**Scilab code Exa 2.13** summation value

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

j=[]
s=[]
i=0
upj=input("Enter the upper limit for the inner summation:");
lowj=input("Enter the lower limit for the inner summation:");
upi=input("Enter the upper limit for the outer summation:");
lowi=input("Enter the lower limit for the outer summation:");
for i=lowj:upj+1
    j=j+1
end
for l=lowi:upi+1
    s=s+(j*l)
```

```
end
mprintf(”%d”,s)
```

**Scilab code Exa 2.14** Sum of values of set

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

s=0
res=[]
mprintf(”Sum of values of s for all the members of the set { ”)
for s=0:2:4
    mprintf(”%d ”,s)
    res=res+s
end
mprintf(”} is %d”,res)
```

**Scilab code Exa 2.15** Summation k power 2

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

n1=100
n2=49

//From table 2 summation k^2=(n(n+1)(2n+1))/6
```

```
v1=(n1*(n1+1)*(2*n1+1))/6
v2=(n2*(n2+1)*(2*n2+1))/6

v=v1-v2

mprintf("Summation k^2 ,k=50 to 100 is %d",v)
```

**Scilab code Exa 2.16** Sum of Matrices

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

matA=[]
mprintf("Enter the dimensions of MATRIX A:")
row=input("Enter the no. of rows:")
col=input("Entet the no.of columns:")
mprintf("Enter the elements:")
for i=1:row
    for j=1:col
        mprintf('\nInput for Row %d , Column %d:',i,j)
        n=input(" ")
        matA(i)(j)=n
end
end

matB=[]
mprintf("Enter the dimensions of MATRIX B:")
row1=input("Enter the no. of rows:")
col1=input("Entet the no.of columns:")
mprintf("Enter the elements:")
```

```
for i=1:row1
    for j=1:col1
        mprintf('\nInput for Row %d , Column %d:',i,j)
        n=input(" ")
        matB(i)(j)=n
end
end
mprintf("Matrix A:")
disp(matA)
mprintf("Matrix B:")
disp(matB)
matADD=matA+matB
mprintf("Sum of Matrices:")
disp(matADD)
```

**Scilab code Exa 2.17** Matrix property

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

A = [[1,1],
[2,1]]

B = [[2,1],
[1,1]]

m1=A*B
m2=B*A

disp(m1,'A*B=')
disp(m2,'B*A=')
```

```

if m1==m2 then
    disp('AB=BA')
else
    disp('AB!=BA')
end
```

**Scilab code Exa 2.18** multiplication of the two matrices

```
//Chapter 02:Basic Structures: Sets, Functions,
   Sequences, Sums and Matrices

clc;
clear;

X = [[1,0,4],
    [2,1,1],
    [3,1,0],
     [0,2,2]]

Y = [[2,4],
    [1,1],
    [3,0]]

result = X * Y

mprintf("The multiplication of the two matrices XY
   is:")
disp(result)
```

**Scilab code Exa 2.19** Transpose of Matrix

```
//Chapter 02:Basic Structures: Sets, Functions, Sequences, Sums and Matrices

clc;
clear;

mat=[]

row=input("Enter the no. of rows:")
col=input("Entet the no.of columns:")
mprintf("Enter the elements:")
for i=1:row
    for j=1:col
        mprintf('\nInput for Row %d , Column %d:',i,j)
        n=input(" ")
        mat(i)(j)=n
end
end
mprintf("Original Matrix:")
disp(mat)
matt=mat'
mprintf("Transpose of Matrix:")
disp(matt)
```

# Chapter 3

# Algorithms

**Scilab code Exa 3.1** `largest element`

```
//Chapter 03: Algorithms

clc;
clear;

ar=[]
max_v=0
n=input('Enter the number of elements in the finite sequence:')
disp('Enter the elements one after the other!')
for i=1:n
    ar(i)=input(' ')
end
for i=1:n
    if ar(i)>max_v then
        max_v=ar(i)
        end
end
disp(max_v,'The largest element is:')
```

**Scilab code Exa 3.2** Linear Search

```
//Chapter 03: Algorithms

clc;
clear;

//Linear Search is also known as Sequential Search
function []= linearsearch (a ,n , ie )
i =1;
j =0;
for i =1: n
if ( arr(i) == ie )
printf ( "\nElement:%d found at position %d\n " ,ie , i ) ;
j =1;
end
end
if ( j ==0)
disp ( "Element Not Found!") ;
end
endfunction

arr =[1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22]
l=length(arr)
disp (arr , " Given array:" ) ;
linearsearch (arr ,l ,19)        //Note:input format for function is (array,length,element to be searched)
```

**Scilab code Exa 3.3** binarysearch

```
//Chapter 03: Algorithms

clc;
clear;

function []= binarysearch (arr ,n ,i)
last =1;
h=n;
while (last <= h )
mid = int (( last + h ) /2) ;
if ( arr ( mid ) == i )
printf ( "\nElement:%d found at position %d",i ,mid) ;
break ;
else
if ( arr ( mid ) >i )
h = mid -1;
else
last = mid +1;
end
end
end
endfunction

//Note:input array has to be sorted
ar =[1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22]
l=length(ar)
disp (ar , " Given array " ) ;
binarysearch (ar ,l ,19)         //Note:input format for function is (array,length,element to be searched)
```

**Scilab code Exa 3.4** `bubblesort`

```
//Chapter 03: Algorithms
```

```

clc;
clear;

function [ res ]= bubblesort (a , n )
i =1;
j =1;
temp =0;
for i =1: n -1
for j =1: n - i
if ( a ( j ) >a ( j +1) )
temp = a ( j ) ;
a ( j ) = a ( j +1) ;
a ( j +1) = temp ;
end
j = j +1;
end
i = i +1;
end
res = a ;
disp ( res ,”Sorted Array :”) ;
endfunction

 a =[3 2 4 1 5]
 disp (a , ” Given Array ” )
a1 = bubblesort (a ,5)
```

**Scilab code Exa 3.5** `insertionSort`

```
//Chapter 03: Algorithms

clc;
clear;

function result = insertionSort(Arr)
```

```
    for i=2:length(Arr)
        A = Arr(i);
        j = i-1;
        while (j>0 & Arr(j) > A)
            Arr(j+1) = Arr(j);
            j = j-1;
        end
        Arr(j+1) = A;
    end

result = Arr;
endfunction

arr=[3 2 4 1 5]
disp(arr,”Given Array”)
arr_s=insertionSort(arr)
disp(arr_s,”Sorted Array”)
```

# Chapter 4

# Number Theory and Cryptography

**Scilab code Exa 4.1** `quotient and remainder`

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

//To find the quotient and remainder

dividend=101
divisor=11
quotient=int(dividend/divisor) //To find quotient
remainder=modulo(dividend,divisor) //To find remainder
dividend_a=(divisor *quotient)+remainder //To find dividend
mprintf(”The quotient when %d is divided by %d is %d = %d div %d and the remainder is %d = %d mod %d”,dividend,divisor,quotient,dividend,divisor,remainder,dividend,divisor)
```

**Scilab code Exa 4.2** quotient and remainder

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

//To find the quotient and remainder

dividend=-11
divisor=3
quotient=(dividend/divisor) //To find quotient
remainder=pmodulo(dividend,divisor) //To find
    remainder
dividend_a=(divisor*quotient)+remainder //To find
    dividend
mprintf("The quotient when %d is divided by %d is %.
    f = %d div %d and the remainder is %d = %d mod %d
    ",dividend,divisor,quotient,dividend,divisor,
    remainder,dividend,divisor)
```

**Scilab code Exa 4.3** Decimal Equivalent

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

bin=[]
n=input("Enter the length of the binary number:")
dec=0
disp("Enter the digits one by one")
```

```
for i =1:n
    bin(i)=input(" ")
end
for i=1:n
    dec=dec*2+bin(i)
end
disp(dec,"Decimal Equivalent")
```

**Scilab code Exa 4.4** `Octal to Decimal`

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

i=0
oct=input("Enter the octal number:")
tmp=oct
dec=0
while(oct~=0)
    dec=dec+(modulo(oct,10))*(8**(i+0))
    i=i+1
    oct=int(oct/10)
end
disp(dec,'Equivalent Decimal Value:')
```

**Scilab code Exa 4.5** `hexadecimal to decimal`

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

```

```
dec=[]
d=0
i=1
disp(’Please enter input in inverted commas’)
hex=input(”Enter the hexadecimal number:”)
l=length(hex)
hex=strsplit(hex)
cn=0
for i=l:-1:1
    select hex(i)
    case ’A’ then
        d=10
    case ’B’ then
        d=11
    case ’C’ then
        d=12
    case ’D’ then
        d=13
    case ’E’ then
        d=14
    case ’F’ then
        d=15
        case ’a’ then
        d=10
    case ’b’ then
        d=11
    case ’c’ then
        d=12
    case ’d’ then
        d=13
    case ’e’ then
        d=14
    case ’f’ then
        d=15
    else
        d=eval(hex(i))
    end
    dec=dec+ (d) *(16**cn)
    cn=cn+1
end
disp(dec)
```

**Scilab code Exa 4.6** decimal number to octal

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

arr=[]
n=input("Enter the number:")
tn=n
while n~=0
    re=pmodulo(n,8)
    n=int(n/8)
    arr($+1)=re
end
mprintf("The octal equivalent of the decimal number %d is:",tn)
for i=length(arr):-1:1
    mprintf("%d",arr(i))
end
```

**Scilab code Exa 4.7** decimal to hexadecimal

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

function dec_hex(num)
```

```
rem=[]
i=1
len=0
while num >0
    rem(i)=pmodulo(num,16)
    num=int(num/16)
    i=i+1
    len=len+1
end
disp(”Hexadecimal Equivalent:”)
for i=len:-1:1
    select rem(i)
    case 10 then
        disp('A')
    case 11 then
        disp('B')
    case 12 then
        disp('C')
    case 13 then
        disp('D')
    case 14 then
        disp('E')
    case 15 then
        disp('F')
    else
        disp(rem(i))
end
end
endfunction

inp=input(”Enter the decimal number:”)
dec_hex(inp)
```

**Scilab code Exa 4.8** `decimal number to binary`

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

bin_eq=[]
decn=input(”Enter the decimal number:”)
tn=decn
i=1
b=floor(decn/2)
rem=modulo(decn,2)
bin_eq(i)=string(rem(i))
while 2<=b
    decn=b
    i=i+1
    b=floor(decn/2)
    rem=modulo(decn,2)
    bin_eq(i)=string(rem)
end
bin_eq(i+1)=string(b)
bin_eq=eval(bin_eq)
mprintf(”The binary equivalent of the decimal number %d is:”,tn)
for i=length(bin_eq):-1:1
    mprintf(”%d”,bin_eq(i))
end
```

**Scilab code Exa 4.9** sum of binary numbers

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

bin_a=[]
```

```
i=1
rem=0
n1=input("Enter 1st binary number:")
n2=input("Enter 2nd binary number:")
t1=n1
t2=n2
while (n1~=0 | n2~=0)
    bin_a($+i)=modulo((modulo(n1,10)+modulo(n2,10)+rem),2)
    rem=(modulo(n1,10)+modulo(n2,10)+rem)/2
    n1=int(n1/10)
    n2=int(n2/10)
end
if rem ~=0 then
    bin_a($+i)=rem
end
bin_a=int(bin_a)
bin_a=flipdim(bin_a,1)
mprintf("The sum of binary numbers %d and %d is",t1,t2)
disp(bin_a)
```

**Scilab code Exa 4.10** Prime factors

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

function primefactors(n)
    while modulo(n,2) == 0 //To print all the 2s that divide input
        disp('2')
        n=n/2
    end
    for i=3:2:sqrt(n)//increment by 2 so as to obtain odd numbers only
      while modulo(n,i)==0
          disp(i)
          n=n/i
      end
    end
if(n>2) then //to check for prime number
    disp(n)
    end
endfunction

n1=100
n2=641
n3=999
n4=1024
mprintf("Prime factors of %d are:",n1)
disp(primefactors(n1))
mprintf("\nPrime factors of %d are:",n2)
disp(primefactors(n2))
mprintf("\nPrime factors of %d are:",n3)
disp(primefactors(n3))
mprintf("\nPrime factors of %d are:",n4)
disp(primefactors(n4))
```

**Scilab code Exa 4.11** `prime number checking`

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

n=input("Enter the number:")
c=0
for i =2:n-1
```

```
    if modulo(n,i)==0 then
        c=c+1
    end
end
if c==0 then
    mprintf(”%d is a prime number”,n)
else
    mprintf(”%d is not a prime number”,n)
end
```

**Scilab code Exa 4.12** Prime factors

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

function primefactors(n)
    while modulo(n,2) == 0 //To print all the 2s that divide input
        disp(’2’)
        n=n/2
    end
    for i=3:2:sqrt(n)//increment by 2 so as to obtain odd numbers only
        while modulo(n,i)==0
            disp(i)
            n=n/i
        end
    end
if(n>2) then //to check for prime number
    disp(n)
    end
endfunction

```

```
n1=7007
mprintf("Prime factors of %d are:",n1)
disp(primefactors(n1))
```

**Scilab code Exa 4.13** GCD

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

//GCD using recursion
function f=gcd(n,m)
    if (n>=m) & (modulo(n,m)==0) then
        f=m
    else
        f=gcd(m,modulo(n,m))
    end
endfunction

a=input("Number 1:")
b=input("Number 2:")
ann=gcd(a,b)
mprintf("GCD(%d,%d) is:%d",a,b,ann)
```

**Scilab code Exa 4.14** GCD

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

n1=input("Number 1:")
```

```
n2=input("Number 2:")
a=n1
b=n2
while n1 ~=n2
    if n1>n2 then
        n1=n1-n2
    else
        n2=n2-n1
    end
end
mprintf("GCD(%d,%d) is:%d",a,b,n1)
```

**Scilab code Exa 4.15** GCD using euclidean algorithm

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

//To find the GCD using euclidean algorithm

function gcd(a,b)
    x=a
    y=b
    while y ~=0
        r=modulo(x,y)
        x=y
        y=r
    end
mprintf("GCD(%d,%d) = %d",a,b,x)
endfunction

n1=input("Enter 1st Number:")
n2=input("Enter 2nd Number:")
gcd(n1,n2)
```

**Scilab code Exa 4.16** GCD

```
//Chapter 04:Number Theory and Cryptography

clc;
clear all;

//To find the GCD using euclidean algorithm

function gcd(a,b)
    x=a
    y=b
    while y ~=0
        r=modulo(x,y)
        x=y
        y=r
    end
mprintf("GCD(%d,%d) = %d",a,b,x)
endfunction

n1=input("Enter 1st Number:")
n2=input("Enter 2nd Number:")
gcd(n1,n2)
```

# Chapter 5

# Induction and Recursion

**Scilab code Exa 5.1** recursive function

```
//Chapter 05: Induction and Recursion

clc;
clear;

function f = my_f(n)
if n == 0
  f = 3
else
  f = 2* my_f(n-1) +3 //making a recursive call
end
return f
endfunction

for n=0:4
re=my_f(n)
mprintf("The value of f(%d) is %d\n",n,re)
end
```

**Scilab code Exa 5.2** factorial

```
//Chapter 05: Induction and Recursion

clc;
clear;

function fact = my_factorial(n)
if n == 0
  fact = 1
else
  fact = n * my_factorial(n-1)//recursive function call
end
return fact
endfunction

num=input(”Enter the number whose factorial is to be found:”)
f=my_factorial(num)
mprintf(”The factorial of %d is %d”,num,f)
```

**Scilab code Exa 5.3** power

```
//Chapter 05: Induction and Recursion

clc;
clear;

function pow = power(i,n)
if n == 0
  pow = 1
else
  pow = i * power(i,n-1)//recursive function call
end
```

```
return pow
endfunction

n=input("Enter the number whose power is to be found :")
po=input("Enter the power:")
p=power(n,po)
mprintf("%d to the power %d is %d",n,po,p)
```

**Scilab code Exa 5.4** gcd

```
//Chapter 05: Induction and Recursion

clc;
clear;

function res=greatestcommondivisior(a,b)
    if a==0 then
res=b
    else
res=greatestcommondivisior(modulo(b,a),a)
    end
return res
endfunction

num1=input("Enter the first number:")
num2=input("Enter the second number:")
res_gcd=greatestcommondivisior(num1,num2)
mprintf("The gcd of %d , %d is %d",num1,num2,res_gcd)

//By Using the inbuilt function ,that is provided by Scilab
p=[num1,num2]
res=gcd(p)
```

```
mprintf("\nThe gcd of %d , %d is %d",num1,num2,res)
```

**Scilab code Exa 5.5** merge and sort

```
//Chapter 05: Induction and Recursion

clc;
clear;

//Function to merge & sort
function [ a1 ]= mergesort (a ,p , r )
if (p < r )
q = int (( p + r ) /2) ;
a = mergesort (a ,p , q ) ;
a = mergesort (a , q +1 , r ) ;
a = merge (a ,p ,q , r ) ;
else
a1 = a ;
return ;
end
a1 = a ;
endfunction

//Function to merge
function [ a1 ]= merge (a ,p ,q , r )
n1 =q - p +1;
n2 =r - q ;
left = zeros ( n1 +1) ;
right = zeros ( n2 +1) ;
for i =1: n1
left ( i ) = a ( p +i -1) ;
end
for i1 =1: n2
right ( i1 ) = a ( q + i1 ) ;
end
```

```
left ( n1 +1) =111111111;
right ( n2 +1) =111111111;
i =1;
j =1;
k=p;
for k = p : r
if ( left ( i ) <= right ( j ) )
a ( k ) = left ( i ) ;
i = i +1;
else
a ( k ) = right ( j ) ;
j = j +1;
end
end
a1 = a ;
endfunction

arr =[8 2 4 6 9 7 10 1 5 3]
disp(arr,” Given Array:” ) ;
arr_s =mergesort (arr ,1 ,10)
disp(arr_s , ” Sorted Array:” );
```

# Chapter 6

# Counting

**Scilab code Exa 6.1** `assign offices to employees`

```
//Chapter 06: Counting

clc;
clear;

n=2 //no of employees
r=12 //no of office rooms
sanchez=12
patel=11
sol=sanchez*patel

//product rule
mprintf("Total no of ways to assign offices to these
    employees is %d",sol)
```

**Scilab code Exa 6.2** `Chairs labelling`

```
//Chapter 06: Counting
```

```

clc;
clear;

letters=26 //Total no of letters in the english alphabet
post=100 //Total positive no.s not beyond 100
sol=letters*post

//number of chairs to be labelled with an alphabet and an integer using product rule
mprintf("Total number of chairs that can be labelled with an alphabet and an integer is %d",sol)
```

**Scilab code Exa 6.3** `number of ports`

```
//Chapter 06: Counting

clc;
clear;

mc=32 //total no of microcomputers
p=24 //total no of ports in each microcomputer
sol=mc*p

//total number of different ports to a microcomputer in the center are found using product rule
mprintf("Total number of ports is %d",sol)
```

**Scilab code Exa 6.4** `bit strings of length seven`

```
//Chapter 06: Counting

```

```
clc;
clear;

bits=2 //possible bits are either 0 or 1
ns=7 //no of bits in the string (ie). length of the string
sol=bits**ns

// 7 bits are capable of taking either 0 or 1 so by PRODUCT RULE
mprintf(”Total different bit strings of length seven are %d”,sol)
```

**Scilab code Exa 6.5** `number of choices`

```
//Chapter 06: Counting

clc;
clear;

letters=26 //no. of letters in english alphabet
no_of_letters=3 //number of letters
choices=10 //number of choices for each letter
result=1//in order to avoid junk values.  Assigned it to 1.
for i=1:no_of_letters
result=result*letters*choices
end

mprintf(”The total number of choices are %d”,result)
```

**Scilab code Exa 6.6** `number of ways to select students`

```
//Chapter 06: Counting

clc;
clear;

function res=permutation(n,r) //function definition
i=n
res=1
l=(n-r)+1
u=n
for i=l:u //computing the permutation
res=res*i
end
return res
endfunction

a=permutation(5,3)//function call
b=permutation(5,5)//function call

mprintf(”The number of ways to select 3 students from a group of 5 students to line up for a picture is %d”,a)
mprintf(”\nThe number of ways to select 5 students from a group of 5 students to line up for a picture is %d”,b)
```

**Scilab code Exa 6.7** decide the prize winners

```
//Chapter 06: Counting

clc;
clear;

function res=permutation(n,r) //function definition
i=n
```

```
res=1
l=(n-r)+1
u=n
for i=l:u //computing the permutation
res=res*i
end
return res
endfunction

num=input("Enter the number of people:")
perm=input("Enter the number of prizes:")
result=permutation(num,perm)
mprintf("The number of ways to decide the prize winners is %d ",result)
```

**Scilab code Exa 6.8** decide the prize winners

```
//Chapter 06: Counting

clc;
clear;

function res=permutation(n,r) //function definition
i=n
res=1
l=(n-r)+1
u=n
for i=l:u
res=res*i
end
return res
endfunction

num=input("Enter the number of runners:")
perm=input("Enter the number of prizes:")
```

```
result=permutation(num,perm)
mprintf("The number of ways to decide the prize
   winners is %d ",result)
```

**Scilab code Exa 6.9** `ways to decide the path`

```
//Chapter 06: Counting

clc;
clear;

function res=citycal(n) //function definition
i=n
res=1
for i=1:n-1
res=res*i
end
return res
endfunction

num=input("Enter the number of cities:")
result=citycal(num)
mprintf("The number of possible ways to decide the
   path is %d ",result)
```

**Scilab code Exa 6.10** `combinations`

```
//Chapter 06: Counting

clc;
clear;

```

```
function result=combination(n,r) //function definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

num=input(”Enter the number of elements:”)
com=input(”Enter the number of combinations:”)
res=combination(num,com)
mprintf(”The number of combinations are %d ”,res)
```

**Scilab code Exa 6.11** select cards from a standard deck

```
//Chapter 06: Counting

clc;
clear;

function result=combination(n,r) //function definition
i=n
num=1
denominator=1
l=(n-r)+1
```

```
u=n
for i=1:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

//Part A Solution
num=input("Enter the number of cards in the deck(For standard deck n=52):")
com1=input("Enter the number of cards for poker hands determination:")
com2=input("Enter the number of cards to select no of ways:")
res1=combination(num,com1)
mprintf("The number of poker hands of %d cards that can be dealt are %d ",com1,res1)
res2=combination(num,com2)
mprintf("\nThe number of ways to select %d cards from a standard deck are %d ",com2,res1)
```

**Scilab code Exa 6.12** `number of combinations`

```
//Chapter 06: Counting

clc;
clear;

function result=combination(n,r) //function definition
i=n
```

```
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

num=input("Enter the total number of members in a team:")
com=input("Enter the number of players:")
res=combination(num,com)
mprintf("The number of combinations are %d ",res)
```

**Scilab code Exa 6.13** `number of combinations`

```
//Chapter 06: Counting

clc;
clear;

function result=combination(n,r) //function definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
```

```
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

num=input("Enter the number of astronauts:")
com=input("Enter the number of astronauts to be selected:")
res=combination(num,com)
mprintf("The number of combinations are %d ",res)
```

**Scilab code Exa 6.14** combinations for the selected faculties

```
//Chapter 06: Counting

clc;
clear;

function result=combination(n,r) //function definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
```

```
result=num/denominator
return result
endfunction

num1=input("Enter the total number of faculty in Computer Science department:")
com1=input("Enter the number of faculty to be selected for the Computer Science department:")
res1=combination(num1,com1)

mprintf("The number of combinations for the Computer Science department is %d ",res1)

num2=input("Enter the total number of faculty in the Maths department:")
com2=input("Enter the number of faculty to be selected for the Maths department:")
res2=combination(num2,com2)

mprintf("The number of combinations for the Maths department is %d ",res2)

final_res=res1*res2

mprintf("The total number of combinations for the selected faculties is %d",final_res)
```

**Scilab code Exa 6.15** `ways to deal players cards`

```
//Chapter 06: Counting

clc;
clear;

function result=combination(n,r) //function
```

```
definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

fac=1
nc=52//no of cards in a standard deck
num1=input("Enter the number of cards to distribute:")
num2=input("Enter the number of players:")
for i=1:num2
    fac=fac*combination(nc,num1)
    nc=nc-num1
end

mprintf("The total number of ways to deal %d players %d cards each is",num2,num1)
disp(fac)
```

**Scilab code Exa 6.16** `ways to place objects into distinguishable boxes`

```
//Chapter 06: Counting

clc;
```

```
clear;

function result=combination(n,r) //function definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

num1=input("Enter the number of indistinguishable bins:")
num2=input("Enter the number of distinguishable bins:")

//Using formula C(n+r-1,n-1) we obtain

comb=combination(num2+num1-1,num2-1)

mprintf("There are %d number of ways to place %d objects into %d distinguishable boxes",comb,num1,num2)
```

# Chapter 7

# Discrete Probability

**Scilab code Exa 7.1** `Blue Ball Probability`

```
//Chapter 07: Discrete Probability

clc;
clear;

no_blue=4                  //no of blue balls
no_red=5                   //no of red balls

prob_blue=no_blue/(no_red+no_blue)

disp('The probability that a ball chosen at random
    will be blue is:')
disp(prob_blue)
```

**Scilab code Exa 7.2** `Probability that 7 appears`

```
//Chapter 07: Discrete Probability

```

```
clc;
clear;

//For sum to be 7 out of the total 36 equally likely possible outcomes there are 6 outcomes
//(1,6) (2,5) (3,4) (4,3) (5,2) (6,1)

total_outcomes=36             //total no of outcomes
seven_sum_outcome=6             //no of outcomes where sum of numbers appearing on dice is 7

prob_seven=seven_sum_outcome/total_outcomes

disp('Probability that 7 comes when 2 dice are rolled is')
disp(prob_seven)
```

**Scilab code Exa 7.3** Probability that a player wins the prize

```
//Chapter 07: Discrete Probability

clc;
clear;

//Part a
no_four_digits=10**4              //no of ways to choose 4 digits by the product rule

//since only 1 entry is correct and wins the prize , it is inferred that there is only 1 possible way to choose all the digits correctly
no_correctentry=1               //no of ways to choose all 4 digits correctly
prob_winning=no_correctentry/no_four_digits       //probability of player winning the large prize
```

```

disp(,prob_winning,'Probability that a player wins the large prize is')

//Part b
//to win small prize player must correctly choose exactly 3 of 4 digits

no_correctentry=36        //no of ways to choose 4 digits with exactly three of the four being correct
prob_winning=no_correctentry/no_four_digits        //probability of player winning small prize

disp(,prob_winning,'Probability that a player wins the small prize is')
```

**Scilab code Exa 7.4** Probability of a winning combination

```
//Chapter 07: Discrete Probability

clc;
clear;

function result=combination(n,r) //function definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
```

```
denominator=denominator*j
end
result=num/denominator
return result
endfunction

n1=input('Enter the total numbers: ')
n2=input('Enter the amount of numbers to pick correctly to win the prize:')
win=combination(n1,n2)
p_win=1/win
mprintf('The total no of ways to choose %d numbers out of %d number is: %d',n1,n2,win)
mprintf('\nThe probability of a winning combination is')
disp(p_win)
```

**Scilab code Exa 7.5** Probability that 11 4 17 39 23 are drawn

```
//Chapter 07: Discrete Probability

clc;
clear;

total_balls=50                //total no of balls in bin
pr=1
//Part-(A) Sampling without replacement
given_no=5          //11 4 17 39 23
select_ways=1                 //ways in which that particular order can be drawn
n=total_balls-given_no
for i=total_balls:-1:n+1
pr=pr*i
end
```

```
prob=select_ways/pr
disp(prob,'The probability that 11,4,17,39,23 are drawn in that order is')

//Part-(B) Sampling with replacement
total_ways=total_balls**given_no            //5 is the no.of balls ,i.e, 11 4 17 39 23
select_ways=1                        //numbers are drawn in that order
prob=select_ways/total_ways
disp(prob,'The probability that 11,4,17,39,23 are drawn in that order is')
```

**Scilab code Exa 7.6** Probability of at least one 0 in bit string

```
//Chapter 07: Discrete Probability

clc;
clear;

s=2**10            //no of bits -0,1 power sequence ie 10
eb=1                        //for bits are 1
pEb=eb/s                     //probability of event E bar that all the bits are 1
pE=1-pEb                    //probability of event E
disp(pE,'The probability that the bit string will contain at least one 0 bit is')
```

**Scilab code Exa 7.7** Random integer divisible by either 2 or 5

```
//Chapter 07: Discrete Probability

```

```
clc;
clear;

max_integers=100
E1=100/2     //event that random integer is divisible by 2
E2=100/5     //event that random integer is divisible by 5
E1IE2=100/(5*2)     //event that random integer is divisible by 5 and 2
pE1=E1/max_integers                //probability of event E1
pE2=E2/max_integers               //probability of event E2
pE1IE2=E1IE2/max_integers   //probability of event E1IE2

pE1UE2=pE1+pE2-pE1IE2

disp(pE1UE2,'Probability that random integer is divisible by either 2 or 5 is')
```

**Scilab code Exa 7.8** Exactly four heads

```
//Chapter 07: Discrete Probability

clc;
clear;

times=7       //no of times flipped
total_outcomes=2**times          //outcomes power times flipped

function result=combination(n,r) //function definition
```

```
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

reqd_heads=4       //no of heads coming up
ways_heads=combination(times,reqd_heads)
pH=2/3              //biased coin with probability of heads for 1 head
pT=1-pH          //probability of tails is total probability-heads probability
rpH=pH**reqd_heads    //probability of 4 heads outcome
rpT=pT**(times-reqd_heads)    //probability of tails outcome

prob_four_heads=ways_heads*rpH*rpT        //probability of exactly four heads appearing

disp(prob_four_heads,'The probability of exactly four heads appearing is')
```

**Scilab code Exa 7.9** Exactly eight 0 bits

```
//Chapter 07: Discrete Probability
```

```

clc;
clear;

p0=0.9      //prob of bit 0 generation
p1=1-p0   //prob of bit 1 generation
total_bits=10   //total bits generated
reqd_bits=8   //reqd bits out of totalbits generated

function result=combination(n,r) //function
   definition
i=n
num=1
denominator=1
l=(n-r)+1
u=n
for i=l:u //to compute the value of the numerator
num=num*i
end
for j=1:r //to compute the value of the denominator
denominator=denominator*j
end
result=num/denominator
return result
endfunction

//Using theorem 2
prob_eight_0=combination(total_bits,reqd_bits)*((p0)
   **reqd_bits)*((p1)**(total_bits-reqd_bits))

disp(prob_eight_0,'Probability of exactly eight 0
   bits generated is')
```

**Scilab code Exa 7.10** Spam Messages Reject

```
//Chapter 07: Discrete Probability

clc;
clear;

s_total_msg=2000     //spam messages total
spam_msg=250       //occurrence of 'Rolex' in spam
nspam_msg=5       //occurrence of 'Rolex' in not
   know to be spam
ns_total_msg=1000//not spam messages total
threshold=0.9
p=spam_msg/s_total_msg
q=nspam_msg/ns_total_msg
r=p/(p+q)

if r>threshold then
    disp(r,'R=')
    disp('Reject')
end
```

**Scilab code Exa 7.11** Spam Messages Reject

```
//Chapter 07: Discrete Probability

clc;
clear;

spam_msg=2000           //no of spam messages
nspam_msg=1000       //no of messages that are not
   spam
o_msg_spam=400          //occurrence of stock in
   spam
o_msg_nspam=60         //occurrence of stock in non
   spam
o_msg1_spam=200      //occurrence of undervalued in
```

```
spam
o_msg1_nspam=25        //occurrence of undervalued in non spam
threshold=0.9
p1=o_msg_spam/spam_msg
q1=o_msg_nspam/nspam_msg
p2=o_msg1_spam/spam_msg
q2=o_msg1_nspam/nspam_msg

r=(p1*p2)/(p1*p2+q1*q2)

if r>threshold then
    disp(r,'R=')
    disp('Reject')
end
```

**Scilab code Exa 7.12** Expected value on Die

```
//Chapter 07: Discrete Probability

clc;
clear;

X=[1,2,3,4,5,6]            //possible values on a fair die
p=1/6                      //probability for any value to appear when die is rolled
Ex=0
l=length(X)
for i=1:l
    Ex=Ex+p*X(i)
end

disp(Ex,'Expected value of X')
```

**Scilab code Exa 7.13** Heads Appears on Fair Coin

```
//Chapter 07: Discrete Probability

clc;
clear;

times=8     //time flipped
o1=3            //occurrence of 3 heads
o2=2            //occurrence of 2 heads
o3=2            //occurrence of 2 heads
o4=2            //occurrence of 2 heads
o5=1            //occurrence of 1 head
o6=1            //occurrence of 1 head
o7=1            //occurrence of 1 head
o8=0            //occurrence of 0 heads
peo=1/times          //probability of each outcome
Ex=peo*(o1+o2+o3+o4+o5+o6+o7+o8)

disp(Ex,'Expected heads when fair coin is flipped 3 times')
```

**Scilab code Exa 7.14** No of possible chances

```
//Chapter 07: Discrete Probability

clc;
clear;

tot_out=36          //total no of outcomes when 2 dice are rolled
```

```
X=[2,3,4,5,6,7,8,9,10,11,12]          //possible sum of 2 dice
pX2=1/tot_out  //no of possible chances
pX12=pX2  //no of possible chances
pX3=2/tot_out  //no of possible chances
pX11=pX3  //no of possible chances
pX4=3/tot_out  //no of possible chances
pX10=pX4  //no of possible chances
pX5=4/tot_out  //no of possible chances
pX9=pX5  //no of possible chances
pX6=5/tot_out  //no of possible chances
pX8=pX6  //no of possible chances
pX7=6/tot_out  //no of possible chances

Ex=X(1)*pX2+X(2)*pX3+X(3)*pX4+X(4)*pX5+X(5)*pX6+X(6)*pX7+X(7)*pX8+X(8)*pX9+X(9)*pX10+X(10)*pX11+X(11)*pX12

disp(Ex,'Ex=')
```

# Chapter 8

# Advanced Counting Techniques

**Scilab code Exa 8.1** Extended binomial coefficient

```
//Chapter 08: Advanced Counting Techniques

clc;
clear;

//For (-2 3)
u=-2                   //From definition 2
k=3                   //From definition 2
bin_coeff1=(u*(u-1)*(u-k+1))/factorial(k)

//For (1/2 3)
u=1/2                   //From definition 2
k=3                   //From definition 2
bin_coeff2=(u*(u-1)*(u-k+1))/factorial(k)

mprintf("The extended binomial coefficient for (-2 3) is %d",bin_coeff1)
mprintf("\nThe extended binomial coefficient for (1/2 3) is %f",bin_coeff2)
```

**Scilab code Exa 8.2** Total Students

```
//Chapter 08: Advanced Counting Techniques

clc;
clear;

no_cs=25                              //no of students majoring in computer science
no_math=13                          //no of students majoring in mathematics
no_mathcs=8                        //no of students majoring in computer science and mathematics

aub=no_cs+no_math-no_mathcs

mprintf("The total no of students in the class is %d",aub)
```

**Scilab code Exa 8.3** Positive integers below 1000 divisible by 7

```
//Chapter 08: Advanced Counting Techniques

clc;
clear;

A=int(1000/7)               //set of positive integers not exceeding 1000 and divisible by 7 Note: inferred from Example 2 of Section 4.1
B=int(1000/11)             //set of positive integers not exceeding 1000 and divisible by 11 Note: inferred from Example 2 of Section 4.1
```

```
AIB=int(1000/(7*11))                               //set
   of positive integers not exceeding 1000 and
   divisible by 7 also 11

AUB=A+B-AIB

mprintf("There are %d positive integers not
   exceeding 1000 that are divisible by either 7 or
   11",AUB)
```

**Scilab code Exa 8.4** Total No of Students

```
//Chapter 08: Advanced Counting Techniques

clc;
clear;

no_freshmen=1807;.........//total no if freshmen
no_cs=453;                 //no of students taking
   course in computer science
no_math=567;                //no of students taking
   course in mathematics
no_csmath=299;               //no of students
   taking course in computer science and mathematics

AUB=no_cs+no_math-no_csmath

csmath=no_freshmen-AUB

mprintf("No.of freshmen taking a course in computer
   science or math is %d",AUB)
mprintf("\n No.of freshmen not taking a course in
   either computer science or math is %d",csmath)
```