CHAPTER 3

3.1 (a)

$$(1011001)_2 = (1\times 2^6)+(0\times 2^5)+(1\times 2^4)+(1\times 2^3)+(0\times 2^2)+(0\times 2^1)+(1\times 2^0)$$
$$= 64+16+8+1 = 89$$

(b)

$$(110.00101)_2 = (1\times 2^2)+(1\times 2^1)+(0\times 2^0)+(0\times 2^{-1})+(0\times 2^{-2})+(1\times 2^{-3})+(0\times 2^{-4})+(1\times 2^{-5})$$
$$= 4+2+0.125+0.03125 = 6.15625$$

(c)

$$(0.01011)_2 = (0\times 2^{-1})+(1\times 2^{-2})+(0\times 2^{-3})+(1\times 2^{-4})+(1\times 2^{-5})$$
$$= 0.25+0.0625+0.03125 = 0.34375$$

3.2

$$(71563)_8 = (7\times 8^4)+(1\times 8^3)+(5\times 8^2)+(6\times 8^1)+(3\times 8^0)$$
$$= 28{,}672+512+320+48+3 = 29{,}555$$
$$(3.14)_8 = (3\times 8^0)+(1\times 8^{-1})+(4\times 8^{-2})$$
$$= 3+0.125+0.0625 = 3.1875$$

3.3 Here are VBA and MATLAB implementations of the algorithm:

VBA Procedure	MATLAB M-File

`Option Explicit`
`Sub GetEps()`
`Dim epsilon As Double`
`epsilon = 1`
`Do`
`  If epsilon + 1 <= 1 Then Exit Do`
`  epsilon = epsilon / 2`
`Loop`
`epsilon = 2 * epsilon`
`MsgBox epsilon`
`End Sub`
	`function e = geteps`
	`e = 1;`
	`while(1)`
	`  if e + 1 <= 1, break, end`
	`  e = e / 2;`
	`end`
	`e = 2 * e;`

Both routines yield a result of 2.22044604925031E–16 on my desktop PC. For single precision, the result is 1.192093E–07. Note that MATLAB has a built-in function `eps` that yields the same result.

3.4 Here are VBA and MATLAB implementations of the algorithm:

VBA Procedure	MATLAB M-File

`Option Explicit`
`Sub GetMin()`
`Dim x As Double, xmin As Double`
`x = 1`
`Do`
`  If x <= 0 Then Exit Do`
`  xmin = x`
`  x = x / 2`
`Loop`
`MsgBox xmin`
`End Sub`
	`function xmin = getmin`
	`x = 1;`
	`while(1)`
	`  if x <= 0, break, end`
	`  xmin = x;`
	`  x = x / 2;`
	`end`

Both yield a result of 4.94065645841247E–324 on my desktop PC. For single precision, the result is 1.401298E–45.

3.5 Here is a VBA Program to compute the series in ascending order

```
Option Explicit

Sub SeriesForward()
Dim i As Integer, n As Integer
Dim sum As Single, pi As Double, truth As Double
pi = 4 * Atn(1)
truth = pi ^ 4 / 90
sum = 0
n = 10000
For i = 1 To n
  sum = sum + 1 / i ^ 4
Next i
MsgBox sum
'Display true percent relative error
MsgBox 100 * Abs((truth - sum) / truth)
End Sub
```

This yields a result of 1.0823221 with a true relative error of 1.02838×10^{-4}%.

VBA Program to compute in descending order:

```
Option Explicit

Sub SeriesBackward()
Dim i As Integer, n As Integer
Dim sum As Single, pi As Double, truth As Double
pi = 4 * Atn(1)
truth = pi ^ 4 / 90
sum = 0
n = 10000
For i = n To 1 Step -1
  sum = sum + 1 / i ^ 4
Next i
MsgBox sum
'Display true percent relative error
MsgBox 100 * Abs((truth - sum) / truth)
End Sub
```

This yields a result of 1.0823232 with a true relative error of 3.71×10^{-6}%.

The latter version yields a superior result because summing in descending order mitigates the roundoff error that occurs when adding a large and small number.

3.6 For the first series, after 20 terms are summed, the result is

	A	B	C	D	E	F	G	H
1	x	5						
2	n	n!	x^n/n!	Sign	Series	True Value	et (%)	ea(%)
3	0	1	1	1	1.000000E+00	6.737947E-03	14741.32	
4	1	1	5	-1	-4.000000E+00	6.737947E-03	59465.26	125.0000000
5	2	2	12.5	1	8.500000E+00	6.737947E-03	126051.2	147.0588235
6	3	6	20.83333	-1	-1.233333E+01	6.737947E-03	183142.9	168.9189189
7	4	24	26.04167	1	1.370833E+01	6.737947E-03	203349.7	189.9696049
8	5	120	26.04167	-1	-1.233333E+01	6.737947E-03	183142.9	211.1486486
9	6	720	21.70139	1	9.368056E+00	6.737947E-03	138934.3	231.6530764
10	7	5040	15.50099	-1	-6.132937E+00	6.737947E-03	91120.85	252.7499191
11	8	40320	9.68812	1	3.555184E+00	6.737947E-03	52663.6	272.5068890
12	9	362880	5.382289	-1	-1.827105E+00	6.737947E-03	27216.65	294.5801032
13	10	3628800	2.691144	1	8.640391E-01	6.737947E-03	12723.48	311.4609662
14	11	39916800	1.223247	-1	-3.592084E-01	6.737947E-03	5431.125	340.5397724
15	12	4.79E+08	0.509686	1	1.504780E-01	6.737947E-03	2133.292	338.7115011
16	13	6.23E+09	0.196033	-1	-4.555520E-02	6.737947E-03	776.0992	430.3202153
17	14	8.72E+10	0.070012	1	2.445667E-02	6.737947E-03	262.9692	286.2690254
18	15	1.31E+12	0.023337	-1	1.119380E-03	6.737947E-03	83.38693	2084.8412684
19	16	2.09E+13	0.007293	1	8.412283E-03	6.737947E-03	24.84936	86.6935085
20	17	3.56E+14	0.002145	-1	6.267312E-03	6.737947E-03	6.984846	34.2247480
21	18	6.4E+15	0.000596	1	6.863137E-03	6.737947E-03	1.857988	8.6815321
22	19	1.22E+17	0.000157	-1	6.706341E-03	6.737947E-03	0.469074	2.3380286
23	20	2.43E+18	3.92E-05	1	6.745540E-03	6.737947E-03	0.112692	0.5811105

The result oscillates at first. By $n = 20$ (21 terms), it is starting to converge on the true value. However, the relative error is still a substantial 0.11%. If carried out further to $n = 27$, the series eventually converges to within 7 significant digits.

In contrast the second series converges much faster. It attains 6 significant digits by $n = 20$ with a percent relative error of 8.1×10^{-6}%.

	A	B	C	D	E	F	G	H
1	x	5						
2	n	n!	x^n/n!	Series	1/Series	True Value	et (%)	ea(%)
3	0	1	1	1.000000E+00	1.000000E+00	6.737947E-03	1.47E+04	
4	1	1	5	6.000000E+00	1.666667E-01	6.737947E-03	2.37E+03	5.00E+02
5	2	2	12.5	1.850000E+01	5.405405E-02	6.737947E-03	7.02E+02	2.08E+02
6	3	6	20.83333	3.933333E+01	2.542373E-02	6.737947E-03	2.77E+02	1.13E+02
7	4	24	26.04167	6.537500E+01	1.529637E-02	6.737947E-03	1.27E+02	6.62E+01
8	5	120	26.04167	9.141667E+01	1.093892E-02	6.737947E-03	6.23E+01	3.98E+01
9	6	720	21.70139	1.131181E+02	8.840322E-03	6.737947E-03	3.12E+01	2.37E+01
10	7	5040	15.50099	1.286190E+02	7.774898E-03	6.737947E-03	1.54E+01	1.37E+01
11	8	40320	9.68812	1.383072E+02	7.230283E-03	6.737947E-03	7.31E+00	7.53E+00
12	9	362880	5.382289	1.436895E+02	6.959453E-03	6.737947E-03	3.29E+00	3.89E+00
13	10	3628800	2.691144	1.463806E+02	6.831506E-03	6.737947E-03	1.39E+00	1.87E+00
14	11	39916800	1.223247	1.476038E+02	6.774891E-03	6.737947E-03	5.48E-01	8.36E-01
15	12	4.79E+08	0.509686	1.481135E+02	6.751577E-03	6.737947E-03	2.02E-01	3.45E-01
16	13	6.23E+09	0.196033	1.483096E+02	6.742653E-03	6.737947E-03	6.98E-02	1.32E-01
17	14	8.72E+10	0.070012	1.483796E+02	6.739472E-03	6.737947E-03	2.26E-02	4.72E-02
18	15	1.31E+12	0.023337	1.484029E+02	6.738412E-03	6.737947E-03	6.90E-03	1.57E-02
19	16	2.09E+13	0.007293	1.484102E+02	6.738081E-03	6.737947E-03	1.99E-03	4.91E-03
20	17	3.56E+14	0.002145	1.484124E+02	6.737983E-03	6.737947E-03	5.42E-04	1.45E-03
21	18	6.4E+15	0.000596	1.484130E+02	6.737956E-03	6.737947E-03	1.40E-04	4.01E-04
22	19	1.22E+17	0.000157	1.484131E+02	6.737949E-03	6.737947E-03	3.45E-05	1.06E-04
23	20	2.43E+18	3.92E-05	1.484131E+02	6.737948E-03	6.737947E-03	8.11E-06	2.64E-05

3.7 The true value can be computed as

$$f'(0.577) = \frac{6(0.577)}{(1 - 3 \times 0.577^2)^2} = 2{,}352{,}911$$

Using 3-digits with chopping

$$6x = 6(0.577) = 3.462 \xrightarrow{\text{chopping}} 3.46$$
$$x = 0.577$$
$$x^2 = 0.332929 \xrightarrow{\text{chopping}} 0.332$$
$$3x^2 = 0.996$$
$$1 - 3x^2 = 0.004$$

$$f'(0.577) = \frac{3.46}{(1 - 0.996)^2} = \frac{3.46}{0.004^2} = 216{,}250$$

This represents a percent relative error of

$$\varepsilon_t = \left| \frac{2{,}352{,}911 - 216{,}250}{2{,}352{,}911} \right| = 90.8\%$$

Using 4-digits with chopping

$$6x = 6(0.577) = 3.462 \xrightarrow{\text{chopping}} 3.462$$
$$x = 0.577$$
$$x^2 = 0.332929 \xrightarrow{\text{chopping}} 0.3329$$
$$3x^2 = 0.9987$$
$$1 - 3x^2 = 0.0013$$

$$f'(0.577) = \frac{3.462}{(1 - 0.9987)^2} = \frac{3.462}{0.0013^2} = 2{,}048{,}521$$

This represents a percent relative error of

$$\varepsilon_t = \left| \frac{2{,}352{,}911 - 2{,}048{,}521}{2{,}352{,}911} \right| = 12.9\%$$

Although using more significant digits improves the estimate, the error is still considerable. The problem stems primarily from the fact that we are subtracting two nearly equal numbers in the denominator. Such subtractive cancellation is worsened by the fact that the denominator is squared.

3.8 First, the correct result can be calculated as

$$y = 1.37^3 - 7(1.37)^2 + 8(1.37) - 0.35 = 0.043053$$

(*a*) Using 3-digits with chopping

并不是四舍五入，而是直接把后面的干掉了

$1.37^3 \rightarrow 2.571353 \rightarrow 2.57$
$-7(1.37)^2 \rightarrow -7(1.87) \rightarrow -13.0$
$8(1.37) \rightarrow 10.96 \rightarrow 10.9$
$-\ 0.35$
0.12

This represents an error of

$$\varepsilon_t = \left|\frac{0.043053 - 0.12}{0.043053}\right| = 178.7\%$$

(*b*) Using 3-digits with chopping

$$y = ((1.37 - 7)1.37 + 8)1.37 - 0.35$$
$$y = (-5.63 \times 1.37 + 8)1.37 - 0.35$$
$$y = (-7.71 + 8)1.37 - 0.35$$
$$y = 0.29 \times 1.37 - 0.35$$
$$y = 0.397 - 0.35$$
$$y = 0.047$$

This represents an error of

$$\varepsilon_t = \left|\frac{0.043053 - 0.047}{0.043053}\right| = 9.2\%$$

Hence, the second form is superior because it tends to minimize round-off error.

3.9

20 × 40 × 120 = 96,000 words @ 64 bits/word = 8 bytes/word
96,000 words @ 8 bytes/word = 768,000 bytes
768,000 bytes / 1024 bytes/kilobyte = 750 kilobytes = 0.75 Mbytes

3.10 Here is a MATLAB script to solve the problem. Programs in other languages would have a similar structure and outcome.

```
%  Given: Taylor Series Approximation for
%  cos(x) = 1 - x^2/2! + x^4/4! - ...
%  Find: number of terms needed to represent cos(x) to
%  8 significant figures at the point where: x = 0.3 pi

x = 0.3 * pi;
es = 0.5e-08;
%approximation
cosi = 1;
j = 1;
% j=terms counter
fprintf('j= %2.0f cos(x)= %0.10f\n', j,cosi)
fact = 1;
while(1)
  j = j + 1;
  i = 2 * j - 2;
  fact = fact * i * (i - 1);
  cosn = cosi + ((-1) ^ (j + 1)) * ((x) ^ i) / fact;
  ea = abs((cosn - cosi) / cosn);
```

```
  fprintf('j= %2.0f cos(x)= %0.10f  ea = %0.1e\n',j,cosn,ea)
  if ea < es, break, end
  cosi = cosn;
end

j=  1 cos(x)= 1.0000000000
j=  2 cos(x)= 0.5558678020  ea = 8.0e-001
j=  3 cos(x)= 0.5887433702  ea = 5.6e-002
j=  4 cos(x)= 0.5877699636  ea = 1.7e-003
j=  5 cos(x)= 0.5877854037  ea = 2.6e-005
j=  6 cos(x)= 0.5877852513  ea = 2.6e-007
j=  7 cos(x)= 0.5877852523  ea = 1.7e-009
```

The true value of cos(0.3π) is 0.5877852525. Therefore, 6 terms of the Maclaurin series are necessary to approximate the true value to 8 significant figures.

3.11 First we can evaluate the exact values using the standard formula with double-precision arithmetic as

$$\begin{matrix} x_1 \\ x_2 \end{matrix} = \frac{5{,}000.002 \pm \sqrt{(5{,}000.002)^2 - 4(1)10}}{2(1)} = \begin{matrix} 5{,}000 \\ 0.002 \end{matrix}$$

We can then determine the square root term with 5-digit arithmetic and chopping

$$\sqrt{(5{,}000.0)^2 - 4(1)10} = \sqrt{2{,}500{,}000 - 4(1)10} = \sqrt{24{,}999{,}960} \xrightarrow{\text{chopping}} \sqrt{24{,}999{,}000}$$

$$= 4{,}999.996 \xrightarrow{\text{chopping}} 4{,}999.9$$

Equation (3.12):

$$x_1 = \frac{5{,}000.2 + 4{,}999.9}{2} = \frac{9{,}999.95}{2} \xrightarrow{\text{chopping}} \frac{9{,}999.9}{2} = 4{,}999.95 \xrightarrow{\text{chopping}} 4{,}999.9$$

$$x_2 = \frac{5{,}000.2 - 4{,}999.9}{2} = \frac{0.1}{2} = 0.05$$

Thus, although the first root is reasonably close to the true value (ε_t = 0.002%), the second is considerably off (ε_t = 2400%) due primarily to subtractive cancellation.

Equation (3.13):

$$x_1 = \frac{-2(10)}{-5{,}000.0 + 4{,}999.9} = \frac{-20}{-0.1} = 200$$

$$x_2 = \frac{-2(10)}{-5{,}000.0 - 4{,}999.9} = \frac{-20}{-9{,}999.9} = 0.002$$

For this case, the second root is well approximated, whereas the first is considerably off (ε_t = 96%). Again, the culprit is the subtraction of two nearly equal numbers.

3.12 Remember that the machine epsilon is related to the number of significant digits by Eq. 3.11

$$\xi = b^{1-t}$$

which can be solved in base 10 for a machine epsilon of 1.19209×10^{-7} for

$$t = 1 - \log_{10}(\xi) = 1 - \log_{10}(1.19209 \times 10^{-7}) = 7.92$$

To be conservative, assume that 7 significant figures are good enough. Recall that Eq. 3.7 can then be used to estimate a stopping criterion,

$$\varepsilon_s = (0.5 \times 10^{2-n})\%$$

Thus, for 7 significant digits, the result would be

$$\varepsilon_s = (0.5 \times 10^{2-7})\% = 5 \times 10^{-6}\%$$

The total calculation can be expressed in one formula as

$$\varepsilon_s = (0.5 \times 10^{2-\text{Int}(1-\log_{10}(\xi))})\%$$

It should be noted that iterating to the machine precision is often overkill. Consequently, many applications use the old engineering rule of thumb that you should iterate to 3 significant digits or better.

As an application, I used Excel to evaluate the second series from Prob. 3.5. The results are:

	A	B	C	D	E	F	G	H
1	x	5						
2	n	n!	x^n/n!	Series	1/Series	True Value	et (%)	ea(%)
3	0	1	1	1.000000E+00	1.000000E+00	6.737947E-03	1.47E+04	
4	1	1	5	6.000000E+00	1.666667E-01	6.737947E-03	2.37E+03	5.00E+02
5	2	2	12.5	1.850000E+01	5.405405E-02	6.737947E-03	7.02E+02	2.08E+02
6	3	6	20.83333	3.933333E+01	2.542373E-02	6.737947E-03	2.77E+02	1.13E+02
7	4	24	26.04167	6.537500E+01	1.529637E-02	6.737947E-03	1.27E+02	6.62E+01
8	5	120	26.04167	9.141667E+01	1.093892E-02	6.737947E-03	6.23E+01	3.98E+01
9	6	720	21.70139	1.131181E+02	8.840322E-03	6.737947E-03	3.12E+01	2.37E+01
10	7	5040	15.50099	1.286190E+02	7.774898E-03	6.737947E-03	1.54E+01	1.37E+01
11	8	40320	9.68812	1.383072E+02	7.230283E-03	6.737947E-03	7.31E+00	7.53E+00
12	9	362880	5.382289	1.436895E+02	6.959453E-03	6.737947E-03	3.29E+00	3.89E+00
13	10	3628800	2.691144	1.463806E+02	6.831506E-03	6.737947E-03	1.39E+00	1.87E+00
14	11	39916800	1.223247	1.476038E+02	6.774891E-03	6.737947E-03	5.48E-01	8.36E-01
15	12	4.79E+08	0.509686	1.481135E+02	6.751577E-03	6.737947E-03	2.02E-01	3.45E-01
16	13	6.23E+09	0.196033	1.483096E+02	6.742653E-03	6.737947E-03	6.98E-02	1.32E-01
17	14	8.72E+10	0.070012	1.483796E+02	6.739472E-03	6.737947E-03	2.26E-02	4.72E-02
18	15	1.31E+12	0.023337	1.484029E+02	6.738412E-03	6.737947E-03	6.90E-03	1.57E-02
19	16	2.09E+13	0.007293	1.484102E+02	6.738081E-03	6.737947E-03	1.99E-03	4.91E-03
20	17	3.56E+14	0.002145	1.484124E+02	6.737983E-03	6.737947E-03	5.42E-04	1.45E-03
21	18	6.4E+15	0.000596	1.484130E+02	6.737956E-03	6.737947E-03	1.40E-04	4.01E-04
22	19	1.22E+17	0.000157	1.484131E+02	6.737949E-03	6.737947E-03	3.45E-05	1.06E-04
23	20	2.43E+18	3.92E-05	1.484131E+02	6.737948E-03	6.737947E-03	8.11E-06	2.64E-05
24	**21**	**5.11E+19**	**9.33E-06**	**1.484132E+02**	**6.737947E-03**	**6.737947E-03**	**1.82E-06**	**6.29E-06**
25	22	1.12E+21	2.12E-06	1.484132E+02	6.737947E-03	6.737947E-03	3.91E-07	1.43E-06

Notice how after summing 21 terms, the result is correct to 7 significant figures. At this point, the true and the approximate percent relative errors are at 1.82×10^{-6}% and 6.29×10^{-6}%, respectively. The process would repeat one more time so that the error estimates would fall below the precalculated stopping criterion of 5×10^{-6} %.

3.13 Here are VBA and MATLAB implementations of the algorithm:

VBA Function Procedure	**MATLAB M-File**

Option Explicit
Function SqRoot(a, eps, maxit)
Dim i As Integer, s As Double
Dim sold As Double, ea As Double
i = 0
s = a / 2
ea = 100
Do
  sold = s
  s = (s + a / s) / 2
  i = i + 1
  If s <> 0 Then
  ea = Abs((s - sold) / s) * 100
  End If
  If ea <= eps Or i >= maxit Then Exit Do
Loop
SqRoot = s
End Function
	function [s,ea,i]=SqRoot(a,eps,maxit)
	
	i = 0;
	s = a/2;
	ea = 100;
	while(1)
	  sold = s;
	  s = (s +a/s)/2;
	  i = i + 1;
	  if s ~= 0, ea=abs((s - sold)/s)*100; end
	  if ea <= eps | i >= maxit, break, end
	end