1.

Consider two lists A, B.
eq(A,B) = if A = < > and B = < > then true
else if head(A) = head(B) then eq(tail(A), tail(B))
else false

2.

Let S be a string. Let S^R denote the reverse of S.
pal(S) = if S = < > then true
else if tail(S) = < > then true
else if head(S) = head(S^R) then pal(tail ((tail (S))^R))
else false

3.

a. Let L be a list.
isMember(a, L) = if L = < > then false
else if a = head(L) then true
else isMember(a, tail(L))

b. Let A,B be lists.
isSubset(A,B) = if A = < > then true
else if isMember(head(A), B) then isSubset(tail(A),B)
else false

c. Let A,B be lists.
areEqual(A,B) = if isSubset(A,B) then isSubset(B,A)
else false

d. Let A,B be lists.
union(A,B) = if A = < > then B
else if isMember(head(A),B) then union(tail(A),B)
else union(tail(A), cons(head(A),B))

e. Let A,B be lists.
intersect(A,B) = if A = < > then < >
else if isMember(head(A),B) then head(A):: intersect(tail(A),B)
else intersect(tail(A),B)

f. Let A,B be lists.

```
difference(A,B) = if B = < > then A
                  else if  isMember(head(B),A)
                           then difference(removeAll(head(B),A),tail(B))
                  else difference(A,tail(B))
```

4.

a.

Initial guess	----------	1
sqrt(4,1) = 1:: sqrt(4,2.5)	----------	2.5
sqrt(4,2.5) = 2.5:: sqrt(4,0.5*(2.5 + 4/2.5)) = 2.5::sqrt(4,2.05)	----------	2.05
sqrt(4,2.05) = 2.05:: sqrt(4,.5*(2.05 + 4/2.05)) sqrt(4,2.05) = 2.05:: sqrt(4,2.0006097)	----------	2.0006097

The squre root obtained by a calculator : 2

b.

Initial guess	----------	2
sqrt(4,2) = 2:: sqrt(4,2)	----------	2

You guessed the correct answer.
So no more iterations needed.

The squre root obtained by a calculator : 2

c.

Initial guess	----------	3
sqrt(4,3) = 3:: sqrt(4,2.166666)	----------	2.1666666
sqrt(4,2.1666666) = 2.1666666:: sqrt(4,0.5*(2.1666666 + 4/2.1666666)) = 2.1666666::sqrt(4,2.0064102)	----------	2.0064102
sqrt(4,2.0064102) = 2.0064102:: sqrt(4,.5*(2.0064102 + 4/2.0064102)) = 2.0064102:: sqrt(4,2.0016051)	----------	2.0016051

The squre root obtained by a calculator : 2

d.

Initial guess	----------	1
sqrt(2,1) = 1:: sqrt(2,1.5)	----------	1.5
sqrt(2,1.5) = 1.5:: sqrt(2,0.5*(1.5 + 2/1.5)) = 1.5::sqrt(2,1.4166666)	----------	1.4166666
sqrt(2,1.4166666) = 1.4166666:: sqrt(2,.5*(1.4166666 + 4/1.4166666)) = 1.4166666:: sqrt(2,1.412156)	----------	1.412156

The squre root obtained by a calculator : 1.4142135

e.

Initial guess	----------	1
sqrt(9,1) = 1:: sqrt(9,5)	----------	5
sqrt(9,5) = 5:: sqrt(9,0.5*(5 + 9/5))		
= 5::sqrt(9,3.4)	----------	3.4
sqrt(9,3.4) = 3.4:: sqrt(9,.5*(3.4 + 9/3.4))		
sqrt(9,3.4) = 3.4:: sqrt(9,3.0235294)	----------	3.0235294

The squre root obtained by a calculator : 3

f.

Initial guess	----------	5
sqrt(9,5) = 5:: sqrt(9,3.4)	----------	3.4
sqrt(9,3.4) = 3.4:: sqrt(9,0.5*(3.4 + 9/3.4))		
= 3.4::sqrt(9,3.0235294)	----------	3.0235294
sqrt(9,3.0235294) = 3.0235294:: sqrt(9,.5*(3.0235294 + 9/3.0235294))		
= 3.0235294:: sqrt(9,3.0000915)	----------	3.0000915

The squre root obtained by a calculator : 3

5.

a.

2^31 + 2^27 + 2^26 + 2^25 + 2^24 + 2^15 + 2^11 + 2^0 = 2,399,176,705

b.

2,399,176,705 since unsigned and standard binary are the same.

c.

```
complement   0111 0000 1111 1111 0111 0111 1111 1110
        +                                          1
             ---------------------------------------------------
             0111 0000 1111 1111 0111 0111 1111 1111 = 0x70ff77ff
```

To get the absolute value of the number

7*16^7 + 15*16^5 + 15*16^4 + 7*16^3 + 7*16^2 + 15*16^1 + 15*16
= 1,895,790,591

The string 1000 1111 0000 0000 1000 1000 0000 0001, in 2's complement representation, is the decimal -1,895,790,591

d.

sign : 1 -> negative number
exp : 0001 1110 - 127 = 30 -127 = -97
significand : 1.000 0000 1000 1000 0000 0001
^
| -- the hidden bit

The number is -1.00000001000100000000001*(2^-97)
= -(1 + (2^-8) + (2^-23))*(2^-97)
= -6.3370806*(10^-30)
= -6.3370806e-30

6.

a. Can't represent negative numbers using standard binary.

b. Can't represent with 8 bits

c.

Convert the absolute value (+1996) to standard binary
1996 -> 111 1100 1100
Pad with zero to the left to get a 16 bit representation of the absolute value

0000 0111 1100 1100

Take the 2's complement representation of the absolute value to get the representation of -1996

complement: 1111 1000 0011 0011
\+ 1

1111 1000 0011 0100 (-1996 in 2's comp repre.)

Octal : 1 111 100 000 110 100 -> 0174064
Hexa : 0xf8364

7.

a. $(10)_{10} = (1010)_2$
$(0.0703125)_{10} = (0.0001001)_2$ by the multiplication algorithm

Therefore $(10.0703125)_{10} = (1010.0001001)_2$

b.

1010.0001001 -> 1.0100001001*2^3

8.

$(1.25)_{10}$ -> 1.01 in standard binary ->1.01*2^0 (normalized binary)
Positive number : sign = 0 (bit 31)
exp : 0 + 127 = 127 -> 0111 1111 (bits 30 to 23)

significant : 010 0000 0000 0000 0000 0000 (bits 22 to 0)

Remember that we don't represent the 1 left to the binary point.

So the representation is

```
  0  01111111  01000000000000000000000 = 0x3fa00000
  ^        ^                 ^
sign      exp           significand
```

9.

$(0.125)_{10}$ -> 0.001 in binary. -> 1.0*(2^-3) (normalized binary)

Positive number : sign = 0 (bit 31)
exp : -3 + 127 = 124 -> 0111 1100
significant : 000 0000 0000 0000 0000 0000 (bits 23 to 0)

So the representation is

```
  0  01111100  00000000000000000000000 = 0x3e000000
```