COMP 350 Solutions to Assignment 2

1. (5 points)

```
#include <stdio.h>
//Q1:single precision

int main(void) {
    float x = 1;
    float y = 1;
    float z;

    while (y == 1) {
          x++;
          z = 1/x;
          y = z*x;
     }
    printf("y=%.6e\n",y);
    printf("The smallest positive integer x is %.6e", x);
    return 0;
}
```

Result:

```
y=9.999999e-01
The smallest positive integer x is 4.100000e+01
```

```
#include <stdio.h>
//Q1:double precision

int main(void) {
    double x = 1;
    double y = 1;
    double z;

    while (y == 1) {
          x++;
          z = 1/x;
          y = z*x;
    }
    printf("y=%.6e\n",y);
    printf("The smallest positive integer x is %.6e", x);
    return 0;
}
```

Result:

```
y=1.000000e+00
The smallest positive integer x is 4.900000e+01
```

2. (5 points + 3 bonus points)

```
#include <stdio.h>
#include <math.h>

int main (void) {
    int i;
    float x = 1;

    for (i = 1;i <= 70;i++) {   //i should be large enough to evaluate the limit
        x = 100*x/i;
        printf("x%d=%.6e\n",i,x);
    }
    return 0;
}
```

Result:

```
x1=1.000000e+02
x2=5.000000e+03
x3=1.666667e+05
x4=4.166667e+06
x5=8.333334e+07
x6=1.388889e+09
x7=1.984127e+10
x8=2.480159e+11
x9=2.755732e+12
x10=2.755732e+13
x11=2.505211e+14
x12=2.087676e+15
x13=1.605904e+16
x14=1.147075e+17
x15=7.647165e+17
x16=4.779478e+18
x17=2.811458e+19
x18=1.561921e+20
x19=8.220636e+20
x20=4.110318e+21
x21=1.957294e+22
x22=8.896793e+22
x23=3.868171e+23
x24=1.611738e+24
x25=6.446951e+24
x26=2.479597e+25
x27=9.183691e+25
x28=3.279890e+26
```

```
x29=1.130996e+27
x30=3.769988e+27
x31=1.216125e+28
x32=3.800391e+28
x33=1.151634e+29
x34=3.387158e+29
x35=9.677595e+29
x36=2.688221e+30
x37=7.265461e+30
x38=1.911963e+31
x39=4.902470e+31
x40=1.225617e+32
x41=2.989311e+32
x42=7.117406e+32
x43=1.655211e+33
x44=3.761843e+33
x45=8.359651e+33
x46=1.817315e+34
x47=3.866628e+34
x48=8.055475e+34
x49=1.643975e+35
x50=3.287949e+35
x51=6.446959e+35
x52=1.239800e+36
x53=2.339245e+36
x54=4.331935e+36
x55=inf
x56=inf
x57=inf
x58=inf
x59=inf
x60=inf
x61=inf
x62=inf
x63=inf
x64=inf
x65=inf
x66=inf
x67=inf
x68=inf
x69=inf
x70=inf
```

The final infinity result is due to overflow when single precision is used. To overcome the difficulty, we use double precision, see below.

```
#include <stdio.h>
#include <math.h>

int main (void) {
    int i;
    double x = 1;

    for (i = 1;i <= 300;i++) {   //i should be large enough to evaluate the limit
        x = 100*x/i;
        printf("x%d=%.6e\n",i,x);
    }

    return 0;
}
```

Result:

```
x1=1.000000e+02
x2=5.000000e+03
x3=1.666667e+05
x4=4.166667e+06
x5=8.333333e+07
x6=1.388889e+09
x56=1.406473e+37
x57=2.467496e+37
x58=4.254303e+37
x59=7.210683e+37
x60=1.201780e+38
x61=1.970132e+38
x62=3.177632e+38
x63=5.043861e+38
x64=7.881032e+38
x65=1.212466e+39
x66=1.837070e+39
x67=2.741896e+39
x68=4.032200e+39
x69=5.843769e+39
x70=8.348241e+39
x71=1.175809e+40
x72=1.633067e+40
x73=2.237079e+40
x74=3.023079e+40
x75=4.030772e+40
x76=5.303648e+40
x77=6.887854e+40
x78=8.830583e+40
```

```
x79=1.117795e+41
x80=1.397244e+41
x81=1.724993e+41
x82=2.103650e+41
x83=2.534518e+41
x84=3.017283e+41
x85=3.549745e+41
x86=4.127610e+41
x87=4.744379e+41
x88=5.391340e+41
x89=6.057685e+41
x90=6.730762e+41
x91=7.396441e+41
x92=8.039610e+41
x93=8.644742e+41
x94=9.196534e+41
x95=9.680562e+41
x96=1.008392e+42
x97=1.039579e+42
x98=1.060795e+42
x99=1.071510e+42
x100=1.071510e+42
x101=1.060901e+42
x102=1.040099e+42
x103=1.009805e+42
x104=9.709665e+41
x105=9.247300e+41
x106=8.723868e+41
x107=8.153147e+41
x108=7.549211e+41
x109=6.925881e+41
x110=6.296256e+41
x111=5.672302e+41
x112=5.064556e+41
x113=4.481908e+41
x114=3.931498e+41
x115=3.418694e+41
x116=2.947150e+41
x117=2.518932e+41
x118=2.134688e+41
x119=1.793855e+41
x120=1.494879e+41
x121=1.235438e+41
x122=1.012654e+41
x123=8.232957e+40
x124=6.639481e+40
x125=5.311585e+40
```

```
x126=4.215544e+40
x127=3.319326e+40
x128=2.593223e+40
x129=2.010251e+40
x130=1.546347e+40
x131=1.180417e+40
x132=8.942555e+39
x133=6.723726e+39
x134=5.017706e+39
x135=3.716819e+39
x136=2.732955e+39
x137=1.994858e+39
x138=1.445549e+39
x139=1.039963e+39
x140=7.428310e+38
x141=5.268305e+38
x142=3.710074e+38
x143=2.594457e+38
x144=1.801706e+38
x145=1.242556e+38
x146=8.510659e+37
x147=5.789564e+37
x148=3.911867e+37
x149=2.625414e+37
x150=1.750276e+37
x151=1.159123e+37
x152=7.625811e+36
x153=4.984190e+36
x154=3.236487e+36
x155=2.088056e+36
x156=1.338498e+36
x157=8.525463e+35
x158=5.395862e+35
x159=3.393624e+35
x160=2.121015e+35
x161=1.317401e+35
x162=8.132103e+34
x163=4.989020e+34
x164=3.042086e+34
x165=1.843688e+34
x166=1.110656e+34
x167=6.650632e+33
x168=3.958709e+33
x169=2.342432e+33
x170=1.377901e+33
x171=8.057900e+32
x172=4.684826e+32
```

```
x173=2.707992e+32
x174=1.556317e+32
x175=8.893241e+31
x176=5.052978e+31
x177=2.854790e+31
x178=1.603814e+31
x179=8.959857e+30
x180=4.977698e+30
x181=2.750110e+30
x182=1.511049e+30
x183=8.257100e+29
x184=4.487554e+29
x185=2.425705e+29
x186=1.304142e+29
x187=6.974024e+28
x188=3.709587e+28
x189=1.962744e+28
x190=1.033023e+28
x191=5.408500e+27
x192=2.816927e+27
x193=1.459548e+27
x194=7.523441e+26
x195=3.858175e+26
x196=1.968457e+26
x197=9.992166e+25
x198=5.046548e+25
x199=2.535954e+25
x200=1.267977e+25
x201=6.308343e+24
x202=3.122942e+24
x203=1.538395e+24
x204=7.541153e+23
x205=3.678611e+23
x206=1.785734e+23
x207=8.626732e+22
x208=4.147467e+22
x209=1.984434e+22
x210=9.449686e+21
x211=4.478524e+21
x212=2.112511e+21
x213=9.917894e+20
x214=4.634530e+20
x215=2.155595e+20
x216=9.979608e+19
x217=4.598898e+19
x218=2.109586e+19
x219=9.632813e+18
```

```
x220=4.378551e+18
x221=1.981245e+18
x222=8.924527e+17
x223=4.002030e+17
x224=1.786621e+17
x225=7.940536e+16
x226=3.513511e+16
x227=1.547802e+16
x228=6.788607e+15
x229=2.964457e+15
x230=1.288894e+15
x231=5.579630e+14
x232=2.405013e+14
x233=1.032194e+14
x234=4.411087e+13
x235=1.877058e+13
x236=7.953637e+12
x237=3.355965e+12
x238=1.410069e+12
x239=5.899871e+11
x240=2.458280e+11
x241=1.020033e+11
x242=4.215013e+10
x243=1.734573e+10
x244=7.108906e+09
x245=2.901594e+09
x246=1.179510e+09
x247=4.775344e+08
x248=1.925542e+08
x249=7.733100e+07
x250=3.093240e+07
x251=1.232366e+07
x252=4.890343e+06
x253=1.932942e+06
x254=7.610008e+05
x255=2.984317e+05
x256=1.165749e+05
x257=4.535987e+04
x258=1.758135e+04
x259=6.788165e+03
x260=2.610833e+03
x261=1.000319e+03
x262=3.818011e+02
x263=1.451715e+02
x264=5.498922e+01
x265=2.075065e+01
x266=7.800996e+00
```

```
x267=2.921721e+00
x268=1.090194e+00
x269=4.052768e-01
x270=1.501025e-01
x271=5.538838e-02
x272=2.036337e-02
x273=7.459111e-03
x274=2.722303e-03
x275=9.899285e-04
x276=3.586698e-04
x277=1.294837e-04
x278=4.657686e-05
x279=1.669421e-05
x280=5.962219e-06
x281=2.121786e-06
x282=7.524065e-07
x283=2.658680e-07
x284=9.361550e-08
x285=3.284754e-08
x286=1.148515e-08
x287=4.001796e-09
x288=1.389513e-09
x289=4.808002e-10
x290=1.657932e-10
x291=5.697360e-11
x292=1.951151e-11
x293=6.659217e-12
x294=2.265040e-12
x295=7.678101e-13
x296=2.593953e-13
x297=8.733849e-14
x298=2.930822e-14
x299=9.802079e-15
x300=3.267360e-15
```

Form the new result, we suspect the limit is 0, rather than ∞.

3. (a) (4 points) When n becomes big, 2^{-n} becomes very small, and $2^{-n}x_n$ goes to 0. Then $\sqrt{1+2^{-n}x_n}$ is close to 1. So we will have **cancellation error** when compute $\sqrt{1+2^{-n}x_n}-1$. The computed result will be 0.
The **cancellation error** (difference between the computed result 0 and the true result) is then amplified by 2^{n+1}. So this will cause big numerical problem.
No matter what x_n is, x_{n+1} will always be 0 when n is enough big.

```
#include <stdio.h>
#include <math.h>

int main(void) {
    int i;
    int n = 60;
    double x = 1.0f;
    double p2n = 1.0f;
    double log2 = log(2);

    for (i = 1; i <= n; i++) {
        x = p2n*2*(sqrt(1+1/p2n*x)-1);
        printf("%d: x%d = %.16f   x%d-ln(x0+1) = %.16f\n",
            i, i, x, i, x-log2);
        p2n *= 2;
    }
    return 0;
}
```

```
Result:
1: x1 = 0.8284271247461903   x1-ln(x0+1) = 0.1352799441862450
2: x2 = 0.7568284600108841   x2-ln(x0+1) = 0.0636812794509388
3: x3 = 0.7240618613220615   x3-ln(x0+1) = 0.0309146807621162
4: x4 = 0.7083805188386201   x4-ln(x0+1) = 0.0152333382786748
5: x5 = 0.7007087569317321   x5-ln(x0+1) = 0.0075615763717868
6: x6 = 0.6969143073088304   x6-ln(x0+1) = 0.0037671267488851
7: x7 = 0.6950273424387490   x7-ln(x0+1) = 0.0018801618788037
8: x8 = 0.6940864128518456   x8-ln(x0+1) = 0.0009392322919003
9: x9 = 0.6936165847594111   x9-ln(x0+1) = 0.0004694041994658
10: x10 = 0.6933818297000016   x10-ln(x0+1) = 0.0002346491400563
11: x11 = 0.6932644918933875   x11-ln(x0+1) = 0.0001173113334422
12: x12 = 0.6932058329175561   x12-ln(x0+1) = 0.0000586523576108
13: x13 = 0.6931765059107420   x13-ln(x0+1) = 0.0000293253507967
14: x14 = 0.6931618430280651   x14-ln(x0+1) = 0.0000146624681198
15: x15 = 0.6931545117404312   x15-ln(x0+1) = 0.0000073311804859
16: x16 = 0.6931508461420890   x16-ln(x0+1) = 0.0000036655821437
17: x17 = 0.6931490133574698   x17-ln(x0+1) = 0.0000018327975245
18: x18 = 0.6931480969651602   x18-ln(x0+1) = 0.0000009164052149
19: x19 = 0.6931476388126612   x19-ln(x0+1) = 0.0000004582527159
20: x20 = 0.6931474097073078   x20-ln(x0+1) = 0.0000002291473625
21: x21 = 0.6931472951546311   x21-ln(x0+1) = 0.0000001145946859
```

```
22: x22 = 0.6931472383439541   x22-ln(x0+1) = 0.0000000577840088
23: x23 = 0.6931472104042768   x23-ln(x0+1) = 0.0000000298443316
24: x24 = 0.6931471973657608   x24-ln(x0+1) = 0.0000000168058155
25: x25 = 0.6931471899151802   x25-ln(x0+1) = 0.0000000093552349
26: x26 = 0.6931471824645996   x26-ln(x0+1) = 0.0000000019046543
27: x27 = 0.6931471824645996   x27-ln(x0+1) = 0.0000000019046543
28: x28 = 0.6931471824645996   x28-ln(x0+1) = 0.0000000019046543
29: x29 = 0.6931471824645996   x29-ln(x0+1) = 0.0000000019046543
30: x30 = 0.6931471824645996   x30-ln(x0+1) = 0.0000000019046543
31: x31 = 0.6931471824645996   x31-ln(x0+1) = 0.0000000019046543
32: x32 = 0.6931476593017578   x32-ln(x0+1) = 0.0000004787418125
33: x33 = 0.6931476593017578   x33-ln(x0+1) = 0.0000004787418125
34: x34 = 0.6931457519531250   x34-ln(x0+1) = -0.0000014286068203
35: x35 = 0.6931457519531250   x35-ln(x0+1) = -0.0000014286068203
36: x36 = 0.6931457519531250   x36-ln(x0+1) = -0.0000014286068203
37: x37 = 0.6931457519531250   x37-ln(x0+1) = -0.0000014286068203
38: x38 = 0.6931152343750000   x38-ln(x0+1) = -0.0000319461849453
39: x39 = 0.6931152343750000   x39-ln(x0+1) = -0.0000319461849453
40: x40 = 0.6931152343750000   x40-ln(x0+1) = -0.0000319461849453
41: x41 = 0.6933593750000000   x41-ln(x0+1) = 0.0002121944400547
42: x42 = 0.6933593750000000   x42-ln(x0+1) = 0.0002121944400547
43: x43 = 0.6933593750000000   x43-ln(x0+1) = 0.0002121944400547
44: x44 = 0.6953125000000000   x44-ln(x0+1) = 0.0021653194400547
45: x45 = 0.6953125000000000   x45-ln(x0+1) = 0.0021653194400547
46: x46 = 0.6875000000000000   x46-ln(x0+1) = -0.0056471805599453
47: x47 = 0.6875000000000000   x47-ln(x0+1) = -0.0056471805599453
48: x48 = 0.6875000000000000   x48-ln(x0+1) = -0.0056471805599453
49: x49 = 0.7500000000000000   x49-ln(x0+1) = 0.0568528194400547
50: x50 = 0.7500000000000000   x50-ln(x0+1) = 0.0568528194400547
51: x51 = 1.0000000000000000   x51-ln(x0+1) = 0.3068528194400547
52: x52 = 1.0000000000000000   x52-ln(x0+1) = 0.3068528194400547
53: x53 = 0.0000000000000000   x53-ln(x0+1) = -0.6931471805599453
54: x54 = 0.0000000000000000   x54-ln(x0+1) = -0.6931471805599453
55: x55 = 0.0000000000000000   x55-ln(x0+1) = -0.6931471805599453
56: x56 = 0.0000000000000000   x56-ln(x0+1) = -0.6931471805599453
57: x57 = 0.0000000000000000   x57-ln(x0+1) = -0.6931471805599453
58: x58 = 0.0000000000000000   x58-ln(x0+1) = -0.6931471805599453
59: x59 = 0.0000000000000000   x59-ln(x0+1) = -0.6931471805599453
60: x60 = 0.0000000000000000   x60-ln(x0+1) = -0.6931471805599453
```

(b) (6 points) We may transform the formula for x_{n+1} first.

$$\begin{aligned}
x_{n+1} &= 2^{n+1}(\sqrt{1+2^{-n}x_n}-1) \\
&= 2^{n+1}(\sqrt{1+2^{-n}x_n}-1) * \frac{\sqrt{1+2^{-n}x_n}+1}{\sqrt{1+2^{-n}x_n}+1} \\
&= \frac{2^{n+1}[(\sqrt{1+2^{-n}x_n})^2-1^2]}{\sqrt{1+2^{-n}x_n}+1} \\
&= \frac{2^{n+1}(1+2^{-n}x_n-1)}{\sqrt{1+2^{-n}x_n}+1} \\
&= \frac{2x_n}{\sqrt{1+2^{-n}x_n}+1}
\end{aligned}$$

Use this formula, we can avoid cancellation error.

```
#include <stdio.h>
#include <math.h>

int main(void) {
    int i;
    int n = 60;
    double x = 1.0f;
    double p2n = 1.0f;
    double log2 = log(2);

    for (i = 1; i <= n; i++) {
        x = 2*x/(sqrt(1+p2n*x)+1);
        printf("%d: x%d = %.16f   x%d-ln(x0+1) = %.16f\n",
            i, i, x, i, x-log2);
        p2n /= 2;
    }

    return 0;
}
```

```
Result:
1: x1 = 0.8284271247461901   x1-ln(x0+1) = 0.1352799441862448
2: x2 = 0.7568284600108842   x2-ln(x0+1) = 0.0636812794509389
3: x3 = 0.7240618613220612   x3-ln(x0+1) = 0.0309146807621159
4: x4 = 0.7083805188386214   x4-ln(x0+1) = 0.0152333382786761
5: x5 = 0.7007087569317337   x5-ln(x0+1) = 0.0075615763717884
6: x6 = 0.6969143073088294   x6-ln(x0+1) = 0.0037671267488841
7: x7 = 0.6950273424387612   x7-ln(x0+1) = 0.0018801618788159
8: x8 = 0.6940864128518364   x8-ln(x0+1) = 0.0009392322918911
9: x9 = 0.6936165847594015   x9-ln(x0+1) = 0.0004694041994562
10: x10 = 0.6933818296999493   x10-ln(x0+1) = 0.0002346491400040
11: x11 = 0.6932644918933770   x11-ln(x0+1) = 0.0001173113334317
12: x12 = 0.6932058329179387   x12-ln(x0+1) = 0.0000586523579934
```

```
13: x13 = 0.6931765059118140   x13-ln(x0+1) = 0.0000293253518687
14: x14 = 0.6931618430291043   x14-ln(x0+1) = 0.0000146624691590
15: x15 = 0.6931545117428319   x15-ln(x0+1) = 0.0000073311828866
16: x16 = 0.6931508461384656   x16-ln(x0+1) = 0.0000036655785203
17: x17 = 0.6931490133459748   x17-ln(x0+1) = 0.0000018327860295
18: x18 = 0.6931480969521525   x18-ln(x0+1) = 0.0000009163922072
19: x19 = 0.6931476387558471   x19-ln(x0+1) = 0.0000004581959018
20: x20 = 0.6931474096578458   x20-ln(x0+1) = 0.0000002290979005
21: x21 = 0.6931472951088831   x21-ln(x0+1) = 0.0000001145489378
22: x22 = 0.6931472378344111   x22-ln(x0+1) = 0.0000000572744658
23: x23 = 0.6931472091971775   x23-ln(x0+1) = 0.0000000286372323
24: x24 = 0.6931471948785614   x24-ln(x0+1) = 0.0000000143186161
25: x25 = 0.6931471877192534   x25-ln(x0+1) = 0.0000000071593081
26: x26 = 0.6931471841395995   x26-ln(x0+1) = 0.0000000035796542
27: x27 = 0.6931471823497726   x27-ln(x0+1) = 0.0000000017898273
28: x28 = 0.6931471814548591   x28-ln(x0+1) = 0.0000000008949138
29: x29 = 0.6931471810074024   x29-ln(x0+1) = 0.0000000004474571
30: x30 = 0.6931471807836740   x30-ln(x0+1) = 0.0000000002237287
31: x31 = 0.6931471806718098   x31-ln(x0+1) = 0.0000000001118645
32: x32 = 0.6931471806158777   x32-ln(x0+1) = 0.0000000000559324
33: x33 = 0.6931471805879117   x33-ln(x0+1) = 0.0000000000279664
34: x34 = 0.6931471805739287   x34-ln(x0+1) = 0.0000000000139834
35: x35 = 0.6931471805669371   x35-ln(x0+1) = 0.0000000000069919
36: x36 = 0.6931471805634414   x36-ln(x0+1) = 0.0000000000034961
37: x37 = 0.6931471805616936   x37-ln(x0+1) = 0.0000000000017483
38: x38 = 0.6931471805608197   x38-ln(x0+1) = 0.0000000000008744
39: x39 = 0.6931471805603827   x39-ln(x0+1) = 0.0000000000004374
40: x40 = 0.6931471805601642   x40-ln(x0+1) = 0.0000000000002189
41: x41 = 0.6931471805600550   x41-ln(x0+1) = 0.0000000000001097
42: x42 = 0.6931471805600004   x42-ln(x0+1) = 0.0000000000000551
43: x43 = 0.6931471805599730   x43-ln(x0+1) = 0.0000000000000278
44: x44 = 0.6931471805599594   x44-ln(x0+1) = 0.0000000000000141
45: x45 = 0.6931471805599526   x45-ln(x0+1) = 0.0000000000000073
46: x46 = 0.6931471805599493   x46-ln(x0+1) = 0.0000000000000040
47: x47 = 0.6931471805599476   x47-ln(x0+1) = 0.0000000000000023
48: x48 = 0.6931471805599467   x48-ln(x0+1) = 0.0000000000000014
49: x49 = 0.6931471805599463   x49-ln(x0+1) = 0.0000000000000010
50: x50 = 0.6931471805599461   x50-ln(x0+1) = 0.0000000000000008
51: x51 = 0.6931471805599460   x51-ln(x0+1) = 0.0000000000000007
52: x52 = 0.6931471805599460   x52-ln(x0+1) = 0.0000000000000007
53: x53 = 0.6931471805599460   x53-ln(x0+1) = 0.0000000000000007
54: x54 = 0.6931471805599460   x54-ln(x0+1) = 0.0000000000000007
55: x55 = 0.6931471805599460   x55-ln(x0+1) = 0.0000000000000007
56: x56 = 0.6931471805599460   x56-ln(x0+1) = 0.0000000000000007
57: x57 = 0.6931471805599460   x57-ln(x0+1) = 0.0000000000000007
58: x58 = 0.6931471805599460   x58-ln(x0+1) = 0.0000000000000007
59: x59 = 0.6931471805599460   x59-ln(x0+1) = 0.0000000000000007
```

```
60: x60 = 0.6931471805599460   x60-ln(x0+1) = 0.0000000000000007
```