Ian Benlolo, ID 260744397-Comp 350
Prof.Chang Monday October 2^{nd} 2017

1 Question 1

Here are some of my inputs/outputs. See appendix for code.

```
> run computeProduct 1e-30 3e-38 5e-4
1.5000002E-38 times 10 to the power -33
> run computeProduct 1 10 20 10 10
20000.0 times 10 to the power 0
> run computeProduct 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1.0 times 10 to the power 0
> run computeProduct 3e20 3e25 3e37 3e30
8.1000034E37 times 10 to the power 76
> run computeProduct 1e-30 1e-30 1e-30 1e-30
9.999999E-38 times 10 to the power -83
> |
```

My test results all worked well. The last one outputs $9.99999e-113$ when it should be $1e-113$. This happens because of rounding in IEEE single format.

2 Question 2

2.1 A)

My program (see appendix 2a for code, also attached with this doc in a .java file) gave the following results:

```
> run q2a
 For n= 1 : 0.135279944186245
 For n= 2 : 0.06368127945093882
 For n= 3 : 0.03091468076211623
 For n= 4 : 0.01523333827867479
 For n= 5 : 0.007561576371786782
 For n= 6 : 0.00376712674888513
 For n= 7 : 0.0018801618788036878
 For n= 8 : 9.392322919002938E-4
 For n= 9 : 4.6940419946583933E-4
 For n= 10 : 2.3464914005633108E-4
 For n= 11 : 1.173113334421716E-4
 For n= 12 : 5.8652357610822214E-5
 For n= 13 : 2.932535079669396E-5
 For n= 14 : 1.4662468119763794E-5
 For n= 15 : 7.33118048590331E-6
 For n= 16 : 3.665582143708157E-6
 For n= 17 : 1.8327975245258088E-6
 For n= 18 : 9.164052149346347E-7
 For n= 19 : 4.582527158847327E-7
 For n= 20 : 2.29147362529325E-7
 For n= 21 : 1.1459468585162114E-7
 For n= 22 : 5.778400880007695E-8
 For n= 23 : 2.984433156161259E-8
 For n= 24 : 1.6805815516995892E-8
 For n= 25 : 9.355234920072064E-9
 For n= 26 : 1.904654323148236E-9
 For n= 27 : 1.904654323148236E-9
 For n= 28 : 1.904654323148236E-9
 For n= 29 : 1.904654323148236E-9
 For n= 30 : 1.904654323148236E-9
 For n= 31 : 1.904654323148236E-9
 For n= 32 : -4.7493250387997676E-7
 For n= 33 : -1.4286068202862268E-6
 For n= 34 : -1.4286068202862268E-6
 For n= 35 : -1.4286068202862268E-6
 For n= 36 : -1.4286068202862268E-6
 For n= 37 : -1.4286068202862268E-6
 For n= 38 : -3.194618494528623E-5
 For n= 39 : -3.194618494528623E-5
 For n= 40 : -3.194618494528623E-5
 For n= 41 : -2.760868099452862E-4
 For n= 42 : -7.643680599452862E-4
 For n= 43 : -0.0017409305599452862
 For n= 44 : -0.0017409305599452862
 For n= 45 : -0.005647180559945286
 For n= 46 : -0.005647180559945286
 For n= 47 : -0.005647180559945286
 For n= 48 : -0.005647180559945286
 For n= 49 : -0.06814718055994529
 For n= 50 : -0.1931471805599453
 For n= 51 : -0.1931471805599453
 For n= 52 : -0.6931471805599453
 For n= 53 : -0.6931471805599453
 For n= 54 : -0.6931471805599453
 For n= 55 : -0.6931471805599453
 For n= 56 : -0.6931471805599453
 For n= 57 : -0.6931471805599453
 For n= 58 : -0.6931471805599453
 For n= 59 : -0.6931471805599453
 For n= 60 : -0.6931471805599453
```

The problem here is that it converges to $-ln(2)$ when it should be converging to 0 since we our formula converges to $ln(2)$ so $ln(2) - ln(2)$ should be 0 but it converges to $-ln(2)$. This happens due to discretization errors as well as cancellation errors. A computer can only do a finite number of evaluations while the function is continuous, and not discrete.

2.2 B)

$$x_{n+1} = 2^{n+1}(\sqrt{1+2^{-n}x_n} - 1) \times \frac{\sqrt{1+2^{-n}x_n}+1}{\sqrt{1+2^{-n}x_n}+1} = \ldots = \frac{2x_n}{\sqrt{1+2^{-n}x_n}+1}$$

Now, computing $x_n - ln(2)$ gives the following results:

```
For n= 1 : 0.1352799441862449
For n= 2 : 0.06368127945093915
For n= 3 : 0.03091468076211612
For n= 4 : 0.015233338278676234
For n= 5 : 0.007561576371788559
For n= 6 : 0.003767126748884353
For n= 7 : 0.0018801618788161223
For n= 8 : 9.3923229189119E-4
For n= 9 : 4.694041994562914E-4
For n= 10 : 2.3464914000403958E-4
For n= 11 : 1.173113334317355E-4
For n= 12 : 5.8652357993294046E-5
For n= 13 : 2.932535186850327E-5
For n= 14 : 1.4662469158932545E-5
For n= 15 : 7.3311828865385564E-6
For n= 16 : 3.6655785201622493E-6
For n= 17 : 1.8327860293876341E-6
For n= 18 : 9.163922070065667E-7
For n= 19 : 4.5819590155371515E-7
For n= 20 : 2.290979003172211E-7
For n= 21 : 1.1454893755757922E-7
For n= 22 : 5.727446561465399E-8
For n= 23 : 2.8637232030170878E-8
For n= 24 : 1.4318615737529683E-8
For n= 25 : 7.15930781325369E-9
For n= 26 : 3.579653906626845E-9
For n= 27 : 1.7898269533134226E-9
For n= 28 : 8.949134766567113E-10
For n= 29 : 4.4745673832835564E-10
For n= 30 : 2.2372836916417782E-10
For n= 31 : 1.1186407355978645E-10
For n= 32 : 5.5932147802195686E-11
For n= 33 : 2.7966184923400306E-11
For n= 34 : 1.3983147972851384E-11
For n= 35 : 6.991629497576923E-12
For n= 36 : 3.495870259939693E-12
For n= 37 : 1.748046152272309E-12
For n= 38 : 8.741896095898483E-13
For n= 39 : 4.3720582709738665E-13
For n= 40 : 2.1860291354869332E-13
For n= 41 : 1.0935696792557792E-13
For n= 42 : 5.473399511402022E-14
For n= 43 : 2.731148640577885E-14
For n= 44 : 1.3766765505351941E-14
For n= 45 : 6.994405055138486E-15
For n= 46 : 3.6637359812630166E-15
For n= 47 : 1.9984014443252818E-15
For n= 48 : 1.1102230246251565E-15
For n= 49 : 7.771561172376096E-16
For n= 50 : 4.440892098500626E-16
For n= 51 : 4.440892098500626E-16
For n= 52 : 4.440892098500626E-16
For n= 53 : 4.440892098500626E-16
For n= 54 : 4.440892098500626E-16
For n= 55 : 4.440892098500626E-16
For n= 56 : 4.440892098500626E-16
For n= 57 : 4.440892098500626E-16
For n= 58 : 4.440892098500626E-16
For n= 59 : 4.440892098500626E-16
For n= 60 : 4.440892098500626E-16
```

Which effectively converges to 0 as we hoped.

3 Appendix

3.1 1

```
import java.util.*;

public class computeProduct{
  public static void main(String[] args){
   int count=args.length;
   ArrayList<String> evenArg=new
       ArrayList<String>(Arrays.asList(args));

   if(count%2!=0){ //we will iterate through args in 2's so to make
       sure we
     evenArg.add(0,"1"); //have an even number of args and if we
         dont
     count++;       // add a "1" at the end which wouldnt change
         anything
   }
   int k=0; //the value of K
   float finl=1; //the final result well will be printing


   for (int i=0; i<count-1; i+=2){ //iterate through every inputted
       number
     float x= Float.parseFloat(evenArg.get(i)); float
         y=Float.parseFloat(evenArg.get(i+1));
     float temp;//start of by parsing args to floats

     while (doesOverflow(x,y)){ //if the multiplication would
         overflow
       k++;
       if(x >= Float.MIN_NORMAL*10) x/=10; //to account for
           underflowing of x
       else y/=10; //wouldnt underflow because its x*y is
           overflowing..
     }
     while(doesUnderflow(x,y)){
       k--;
       if(x <= Float.MAX_VALUE/10) x*=10;
```

```
      else y*=10;
    }
    temp=x*y; //this does not overflow or underflow
    if(doesOverflow(temp, finl) || doesUnderflow(temp, finl)){
        //we will do the same as before
      while(doesOverflow(temp, finl)){//just now we add x and y to
          'finl'
        k++;
        if(temp>=Float.MIN_NORMAL*10) temp/=10;
        else finl/=10;
        }
      while (doesUnderflow(temp, finl)){
        k--;
        if(temp<=Float.MAX_VALUE/10) temp*=10;
        else finl*=10;
        }
      }
    finl*=temp; //wont over/under flow at this point!
    }
   System.out.println(""+finl+" times 10 to the power "+k);
  }
  public static boolean doesOverflow(float x, float y){ //simply
      checks if multiplying
    if (x*y >= Float.MAX_VALUE) return true; //these two numbers
        will overflow
    else return false;
  }
  public static boolean doesUnderflow(float x, float y){ //same but
      for underflowing
    if (x*y<=Float.MIN_NORMAL)return true;
    else return false;
  }
}
```

3.2 2

```
import java.math.*;
```

```
public class q2a{
  public static void main(String[] args){
    //Dynamic Programming method
    double[] x=new double[61];
    x[0]=1; //this is x_0
    for(int i=0; i<60; i++){
      x[i+1] = Math.pow(2,i+1)*(Math.sqrt(1+Math.pow(2,-i)*x[i])-1);
    } //we just stored every value into an array using dynamic
        programming


    for(int i=1; i<61; i++){
      double y=x[i]-Math.log(2); // since x_0 is 1 its just ln(2)

      System.out.println(" For n= "+i+" : "+y);

    }

    double[] z=new double[61];    //q2b
    z[0]=1;
    for(int i=0; i<60; i++){
      z[i+1]=2*z[i]/(Math.sqrt(1+Math.pow(2,-i)*z[i])+1);

    }
    System.out.println("q2b");
     for(int i=1; i<61; i++){
      double y=z[i]-Math.log(2);

      System.out.println(" For n= "+i+" : "+y);

    }
  }
}
```