1. (to be completed by 12:20pm) Let $0 < h$ and for $n \geq 1$ let

$$a_n = \frac{h^n}{n!}.$$

Show that for any $\epsilon > 0$ there is a $N(\epsilon, h) \equiv N \geq 1$ such that

$$0 < a_n < \epsilon, \quad \text{for all } n \geq N$$[1]

2. The below R-code is available in Dropbox shared folder (bp.R)

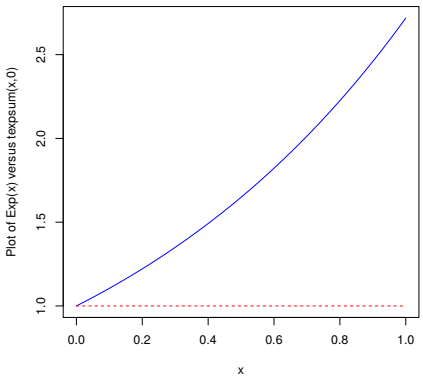
```
> bitsOfPrecision = function(x)max(which( x !=  x*(1+2^-(1:60))))
> bitsOfPrecision(.Machine$double.xmin/2)
> # found at
> # https://r.789695.n4.nabble.com/
> # double-xmin-really-the-smallest-non-zero-normalized-floating-point-number-td4675820.html
```

Try to understand how R stores numbers below .Machine$double.xmin.

3. The below R-code is available in Dropbox shared folder (exptaylor.R)

```
> texpsum = function(x,n= 1500){
+ # texpsum : evaluates taylor polynomial of exp upto degree n at x
+ #
+ # Synopsis: texpsum(x)
+ #           texpsum(x,n)
+ #
+ # Input: x   = argument
+ #        n   = (optional) maximum number of terms. Default: n = 1500
+ #
+ # Output:  texpsum = value of nth degree taylor polynomial at x
+ term = 1
+  ssuma = term
+ m= n+1
+ for(k in 2:m){
+   term = term*x/(k-1)              # Next term in the series
+   ssuma = ssuma + term
+ }
+
+ return(ssuma)
+ }
```

Using the above code, plot the taylor polynomial of degree $n = 2, 4, 6, 8, 10$ along with the true exponential function. For instance on the right is a plot of the taylor polynomial of degree $n = 0$ along with the exponential function.




[1] While identifying $N$ find the best possible $N$.

4. The below R-code is available in Dropbox shared folder (expseries.R)

```
> expseries = function(x,tol= 5e-9,n= 1500){
+ # expSeries Finds a taylor polynomial that approximates taylor series for exp.
+ #
+ # Synopsis: expseries(x)
+ #           expseries(x,tol)
+ #           expseries(x,tol,n)
+ #
+ # Input: x   = argument of the exp function, i.e., compute exp(x)
+ #       tol = (optional) tolerance on accumulated sum. Default: tol = 5e-9
+ #
+ #        n   = (optional) maximum number of terms. Default: n = 1500
+ #
+ # Output:  ssum = value of taylor polynomial of degree n or tolerance is met.
+ #             Calculation is terminated when T_k/S_k < tol, where T_k is the
+ #             kth term in the polynomial and
+ # S_k is value of taylor polynomial of degree k
+
+ term = 1
+ ssum = term
+ Eabs=  c()
+ Eabs[1] = abs(ssum-exp(x)) # Initialize
+ for(k in 2:n){
+   term = term*x/(k-1)              # Next term in the series
+   ssum = ssum + term
+   Eabs[k] = abs(ssum-exp(x))
+   if(abs(term/ssum)<tol) break
+   }
+ }
```

Below is a tabulation of results at x=1 for the approximation to the exponential function and a plot of how the absolute difference decreases. Using the above code create a similar output of results at x=10, x=-10 and comment on the differences; along with how you reduce the error.

```
Series approximation to exponential at x = 1
  k    kth-Term  kth-Pol   Abs Diff
  1 1.000000e+00 1.718282 1.000000e+00
  2 1.000000e+00 2.000000 7.182818e-01
  3 5.000000e-01 2.500000 2.182818e-01
  4 1.666667e-01 2.666667 5.161516e-02
  5 4.166667e-02 2.708333 9.948495e-03
  6 8.333333e-03 2.716667 1.615162e-03
  7 1.388889e-03 2.718056 2.262729e-04
  8 1.984127e-04 2.718254 2.786021e-05
  9 2.480159e-05 2.718279 3.058618e-06
 10 2.755732e-06 2.718282 3.028859e-07
 11 2.755732e-07 2.718282 2.731266e-08
 12 2.505211e-08 2.718282 2.260552e-09
 13 2.087676e-09 2.718282 1.728764e-10
Truncation error after 13 terms is
1.728764e-10
```

Plot of Absolute Differences