Math 226A
Homework 1
Due Friday, October 13th

1. Suppose that you are working with a limited computer that only supports addition, subtraction, multiplication, division, rounding, and computing integer powers of numbers. To add more functionality to your computer, you would like to add a function for computing the exponential function e^x using only the arithmetic operations provided. Recall from calculus that, for any real (or complex) value of x, the exponential function $f(x) = e^x$ can be represented by the infinite series

$$e^x = \lim_{n\to\infty} \sum_{j=0}^{n} \frac{x^j}{j!},$$

which converges for $-\infty < x < \infty$. Since you learned calculus because you thought it would be useful, you might expect to use this series in your algorithm for computing e^x.

   (a) Create a function (using for example MATLAB) that computes e^x by truncating the series above to n terms. Using your function for $n = 10$, compute an approximation to e^x for $x = \pm 0.5$ and $x = \pm 30\pi$, and report the relative error in your approximation. Repeat these computations for $n = 40$. Comment on your results.

   (b) By exploiting properties of the exponential, design an algorithm for accurately computing the value of e^x using the series above, for a wide range of x (e.g. $-100 \le x \le 100$). This algorithm can only use the basic operations of the computer described above. Create a function for your algorithm and use it, with $n = 10$, for computing approximations to e^x for $x = \pm 0.5$ and $x = \pm 30\pi$. Report the relative error in these approximations. If done correctly, your algorithm should produce approximations for these values of x with relative errors $< 10^{-13}$.

   > Hint: Based on the remainder theorem for Taylor series, for what values of x do you expect the above series to be the most accurate? Exploit properties of the exponential so that your algorithm only uses the series approximation for e^x for this range of x.

   Actual algorithms for computing e^x use a rational function (ratio of two polynomials) approximation instead of a Taylor series approximation or they use an *arithmetic geometric mean (AGM)* iteration.

2. Suppose we wish to evaluate the integral

$$I(x) = \int_x^{x+1} \frac{1}{1+s^2} ds$$

using the analytic result

$$I(x) = \arctan(x+1) - \arctan(x)\ . \tag{1}$$

   (a) Explain why (1) is unacceptable for computations when x is large, e.g. when $x = 10^8$.

   (b) Find some exact way to re-write (1) in an acceptable form for computation when x is large.

   (c) Compare your formula in Part (b) with the formula in (1) for a range of large x up to $x = 10^{10}$.

Hint: When you evaluate $I(x)$ using the formula you derived in Part (b) at $x = 10^8$, you should obtain a value like $I(x) \approx 9.99 \times 10^{-17}$, not $I(x) = 0$.

3. You are running a simulation that updates time every 0.1 seconds. The time in the simulation is kept by incrementing a time variable. See the below code.

```
dt = 0.1;          % time step
t  = 0;            % initialize time
Nsteps = 864000;   % number of steps to take

% loop in time
%
for j=1:Nsteps

   %
   % SOME SIMULATION
   %

   % update time
   %
   t = t + dt;
end
```

   (a) Suppose you are simulating one day (86,400 s). Implement the above code, and compute the absolute and relative errors in the time at the end of the simulation.

   (b) Change the time increment to 0.125 s and again run the simulation to 1 day. What are the relative and absolute error in the time?

   (c) Explain the difference in the results from parts (a) and (b).

4. Let q_{20} be the polynomial

$$q_{20}(x) = \prod_{j=1}^{20} (x - j). \tag{2}$$

We can express the polynomial as

$$q_{20}(x) = \sum_{j=0}^{20} a_j x^j, \tag{3}$$

where the coefficients are given in the table below. In MATLAB , we can find the coefficients (a_j) given the roots using the `poly` command. Given the coefficients the polynomial can be evaluated using the `polyval` command.

   (a) Find all the roots of q_{20} by evaluating the expanded form using MATLAB 's `fzero` (or something similar based on the bisection algorithm) or write your own bisection code. Compute the relative error for each root. Which roots can you get accurate to 10 digits? Which do you get least accurately?

(b) Let p be the polynomial

$$p(x) = \sum_{k=0}^{n} a_k x^k.$$

Show that the condition number of the j^{th} root, r_j, associated with changes in the i^{th} coefficient is

$$\kappa = \frac{|a_i r_j^{i-1}|}{|p'(r_j)|}.$$

(c) For a given root r_j, we will define the condition number of the root with respect to changes in the polynomial coefficients as

$$\kappa = \frac{\max_i |a_i r_j^{i-1}|}{|p'(r_j)|}.$$

For q_{20} compute the condition number of each root. Make a graph or table which shows the condition number for each root and the relative error you found in part (a). Does the condition number give information about your results from part (a)? Discuss.

Table 1: Coefficients of q_{20}

j	a_j
20	1
19	-210
18	20615
17	-1256850
16	53327946
15	-1672280820
14	40171771630
13	-756111184500
12	11310276995381
11	-135585182899530
10	1307535010540395
9	-10142299865511450
8	63030812099294896
7	-311333643161390640
6	1206647803780373360
5	-3599979517947607200
4	8037811822645051776
3	-12870931245150988800
2	13803759753640704000
1	-8752948036761600000
0	2432902008176640000