Programming Assignment 7:
Complex Numbers in Pascal, Using Records
CS 2073, Computer Programming with Engineering Applications
Spring Semester, 1992

This assignment uses Pascal record features to implement complex numbers. Recall that a complex number has the form \( a + b j \), where \( a \) is the real part, \( b \) is the imaginary part, and \( j \) represents the square root of -1. (In mathematics, one often writes \( i \) in place of \( j \).) The usual operations of addition, multiplication and modulus (absolute value) are available for complex numbers, defined as follows:

If \( x = a + b j \) and \( y = c + d j \) are two complex numbers and \( r \) is a real number, then

\[
x + y = (a + c) + (b + d) j, \text{ i.e., the real part is } a + c, \text{ and the imaginary part is } b + d.
\]

\[
x * y = (a*c - b*d) + (a*d + b*c) j, \text{ i.e., the real part is } a*c - b*d, \text{ and the imaginary part is } a*d + b*c.
\]

\[
\text{modulus}(x) = \sqrt{a^2 + b^2}, \text{ which is similar to an absolute value for complex numbers.}
\]

\[
r * x = (r*a) + (r*b) j, \text{ i.e., the real part is } r*a, \text{ and the imaginary part is } r*b.
\]

Your program should read in a complex number (read two real numbers, the real and imaginary parts of the complex number). Then it should use a Pascal function to calculate the function \( \exp (e \text{ to a power}) \), using the formula:

\[
\exp(z) = 1 + z + (1/2!)z^2 + (1/3!)z^3 + (1/4!)z^4 + (1/5!)z^5 + \ldots
\]

where \( z \) is a complex number. This formula works for all complex numbers \( z \). (Your program should keep adding in terms until the modulus of a term is less than 0.000001.)

Use the following declarations for a complex number:

```pascal
type complex = record
  re: real,
  im: real
end;

var x, y, z: complex;
```

You must use Pascal procedures and functions to implement various operations. Sample header declarations follow:
procedure add(x, y: complex; var z: complex); (* adds x to y, giving z *)
procedure mul(x, y: complex; var z: complex); (* mults x and y, giving z *)
function modulus(x: complex): real; (* finds modulus of x *)
procedure realmul(r: real; x: complex; var z: complex); (* mults r and x *)
procedure writecomp(x: complex); (* writes out x and modulus(x) *)
procedure readcomp(var z: complex); (* reads in z *)
procedure compexp(x: complex; var z: complex); (* finds exp(x), giving z *)

Test your routines out first with the following code:

begin (* main program *)
  x.re := 0.5; x.im := 0.866025403; (* sqrt(3)/2 *)
  add(x, x, z);
  writecomp(z); (* 1.00000000 + 1.73205080 j, with modulus: 1.99999999 *)
  realmul(2.0, x, z);
  writecomp(z); (* 1.00000000 + 1.73205080 j, with modulus: 1.99999999 *)
  mul(x, x, z);
  writecomp(z); (* -0.49999999 + 0.86602540 j, with modulus: 0.99999999 *)

Then actually run your program with input the number πj, i.e., 0.0 + 3.14159265 j. (Thus the two numbers read by readcomp will be 0.0 and 3.14159265.) You should add a call to writecomp inside the loop in compexp, so that you can see the complex number produced at each stage. (The answer is somewhat unusual.)