Assignment 2

Problem 1

(a) The following list contains valid and invalid Fortran constants. State whether or not each constant is valid. If the constant is valid, specify its type. If it is invalid, briefly state why it is invalid.

- `'Distance =`
- `0.5e-3`
- `0.5d-3`
- `.true.`
- `.maybe.`
- `2^3`
- `'Don't panic'`
- `"Don't panic"`
- `19,5`
- `"3.14159"`
- `3.14159`
- `10.`
- `"Pass / Fail"`

(b) For each expression below, determine whether it is valid or invalid in Fortran. If valid, predict the value and type of the result. Verify by writing, compiling, and running a Fortran program that executes the expression.

- `5 ** 2 / 3 + 1`
- `3.(-4.  / 2)`
- `(2.)  ** real(nint(-2.2))`
- `int(1.5) ** 2.  / floor(2.2)`

(c) Assume a Fortran program with implicit none and these declarations:

```
integer  :: x, y
real     :: r
```

For each variable assignment below, determine if it is valid or invalid. If valid, say what it evaluates to. If invalid, provide a brief explanation.

- `x*y = 2 + 5`
- `xy = 2 + 5`
- `int(2.)  = x`
- `r = 7 / 2`
- `r = 7.  / 2`

(d) Given the atomic masses m_1 and m_2, the reduced mass μ of a diatomic molecule is

$$\mu = \frac{m_1 m_2}{m_1 + m_2}.$$

Write a Fortran program that reads the masses of the two atoms in atomic mass units, computes the reduced mass μ, and writes out μ in grams. Use meaningful variable names and include comments. What is the reduced mass of the CO molecule?

Problem 2

Many fractions cannot be represented exactly in binary floating-point arithmetic. This exercise explores how the types and precision of the operands affect numerical results. Assume that default `real` arithmetic uses single precision (32 bits) and `double precision` arithmetic uses 64 bits.

(a) Write a Fortran program that evaluates the following three expressions exactly as written. Store the three sums and the reference value in part (b) in separate variables declared as `double precision`.

(i) `1.d0/3.  + 1/3`

(ii) `1./3.  + 1.d0/3.`

(iii) `1.d0/3.  + 1./3.d0`

Print each sum with 15 digits after the decimal point.

Hint: If a sum is stored in a variable named `result`, you can print it using

```
write(*,'(F25.15)') result
```

The format `F25.15` uses a field 25 characters wide with 15 digits after the decimal point.

(b) Compute a reference value for $\frac{2}{3}$ using double-precision division. Store it in the reference variable and print it using the same format as in part (a).

(c) Compare each sum from part (a) with the reference value from part (b). State which stored results are exactly equal to the reference, and explain any differences by considering the types of the operands and the precision used in each operation.

Problem 3

The factorial of a positive integer n is defined as

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1,$$

with the special case

$$0! = 1.$$

Write a Fortran program that asks the user to input an integer n. If $n > 0$, the program should compute $n!$ using a loop. If $n = 0$, the program should return the result $0! = 1$. If $n < 0$, the program should raise an error message and stop execution. Print the result with a clear message, e.g. `Factorial of n is ....`