## Part III - Testing ( 30 Points )

Perform white-box testing analysis of the code given below to generate a set of test cases to thoroughly test the code. Indicate the criteria ( e.g. testing methodology ) you are using to select your test cases.

- double sine( double x, double tolerance, int limit, int & nTerms );

  Uses a Taylor Series expansion about X = 0.0 to estimate the value of sine( x ), in which each term of the series is calculated from the previous term. On input, "tolerance" will be the convergence criteria to terminate calculations, and "limit" will be an upper limit on the number of terms to include in the series. On output, "nTerms" will contain the actual number of terms calculated, and the return value will be the estimated value of sine( x ). In the case of input errors, both "nTerms" and the return value will be set to zero.

Mathematics:

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \cdots$$

Algorithm:

$$term_1 = x$$

$$term_{i+1} = \frac{-\,term_i * x^2}{2i * (\,2i + 1\,)} \qquad \forall\, i > 0$$

Code:

```
double sine( double x, double tolerance, int limit, int & nTerms ){

        if( tolerance < 0.0 || limit < 1 ) {
            nTerms = 0;
            return 0.0;
        }

        double term = x;
        double sum = term;
        int i;

        for( i = 1; i < limit; i++ ) {

            if( fabs( term ) < tolerance )
                break;

            term = -term * x * x / ( 2 * i * ( 2 * i + 1 ) );
            sum += term;

        } // for

        nTerms = i;
        return sum;

} // sine
```

Looking at the 1st if, statement, path, or branch testing would require one case that enters the if block, and one that skips it. More rigorous condition testing calls for all 4 combinations of T & F for the two conditions:

| Case | X | tol | limit | Results if correct |
|---|---|---|---|---|
| TC1 | 2.0 | -0.1 | -1 | 0.0, nterms = ∅ |
| TC2 | 2.0 | -0.5 | 2 | " " |
| TC3 | 2.0 | 0.1 | -3 | " " |
| TC4 | 2.0 | 1.0E-6 | 2 | 0.6667, nterms = 2 |

Looking at the for loop, at least 2 test cases are required - one that exits when the tolerance is reached, and one that exits because the limit is reached. TC 4 already satisfies the latter, so one more needs to be added:

| Case | X | tol | limit | Results if correct |
|---|---|---|---|---|
| TC5 | 2.0 | 1.5 | 10 | 0.667, nterms = 2 |

Boundary testing adds:

| Case | X | tol | limit | Results if correct |
|---|---|---|---|---|
| TC6 | $\pi/6$ | 0.0 | 20 | $\approx$ 0.5, nterms = 20 |
| TC7 | 42.0 | 0.1 | 0 | 0.0, nterms = 0 |
| TC8 | 42.0 | 0.2 | 1 | 42.0, nterms = 1 |
| TC9 | 0.0 | 1.0E-3 | 20 | 0.0, nterms = 1 |

Finally partition/equivalence testing explores X in various quadrants:

| Case | X | tol | limit | Results if correct |
|---|---|---|---|---|
| TC 10 | Ø.Ø5 | 1.0 E-6 | 100 | ~ sine(0.05) |
| TC 11 | $\pi/4$ | ↓ | ↓ | ~ $\sqrt{2}/2$ |
| TC 12 | $\pi/2$ | | | ~ 1.0 |
| TC 13 | $\pi$ | | | ~ Ø.Ø |
| TC 14 | $3\pi/2$ | | | ~ -1.0 |
| TC 15 | $2\pi$ | | | ~ Ø.Ø |
| TC 16 | -1.2 | | | ~ sine(-1.2) |
| TC 17 | 25.0 | | | ~ sine(25.0) |
| TC 18 | -25.0 | | | ~ sine(-25.0) |
| TC 19 | -0.01 | | | ~ sine(-0.01) |
| TC 20 | 42.0 | | | ~ sine(42.0) |

Note: X = 1.0 is <u>not</u> good test input, as it will not identify faults raising X to a power, i.e. multiplying * (X * X)