POLYNOMIAL

Practice Problem

Represent a polynomial of degree *n* in double coefficients

Ex: $2.3\ x^4 - 4\ x + 1.6 = 2.3\ x^4 + 0\ x^3 + 0\ x^2 - 4.0\ x^1 + 1.6\ x^0$

(Here, degree of the polynomial = 4)

```
typedef struct polynomial
  {                               // max degree = 49
        double coef [50]; // coefficients array
        int deg;                 // integer degree
  } Poly;
```

Write a function to read a polynomial

```
void Read (Poly p)
{
    int i;
    printf(“Enter the Degree of
        Polynomial: ”);
    scanf(“%d”,&p.deg);
    for(i=0; i<=p.deg; i=i+1){
        printf(“Enter Coefficient
            for x^%d: ”,i);
        scanf(“%lf”, &p.coef[i]);
    }
}
```

✗

```
Poly Read ()
{
    Poly p;
    int i;
    printf(“Enter the Degree of
        Polynomial: ”);
    scanf(“%d”,&p.deg);
    for(i=0; i<=p.deg; i=i+1){
        printf(“Enter Coefficient for
            x^%d: ”,i);
        scanf(“%lf”, &p.coef[i]);
    }
    return p;
}
```

✓

Write a function to write a polynomial

```
void Write (Poly p)
{
    int i;
    printf(“The Polynomial is:\n”);
    for(i=p.deg; i>=0; i=i-1){
        printf(“ + (%lf) x^%d”, p.coef[i],i);
    }
    printf(“\n”);
}
```

Output (Example Format):

```
    The Polynomial is:
    + (2.300000) x^4 + (0.000000) x^3 + (0.000000) x^2 + (– 4.000000) x^1 + (1.600000) x^0
```

Write a function to evaluate a polynomial

Procedure-1:

```
double Eval1 (Poly p, double x)
{
    int i;
    double result = 0;
    for(i=0; i<=p.deg; i=i+1) {
        result = result + p.coef[i] * pow(x,i);  // calculate x^i every-time
    }
    printf("The Evaluated Value is: %lf\n", result);
    return (result);
}
```

Total no. of multiplications = $1 + 1 + 2 + 3 + . . . + n = 1 + n(n+1)/2$

[assuming pow(x,i) does (i-1) multiplications]

Write a function to evaluate a polynomial

Procedure-2:

```
double Eval2 (Poly p, double x)
{       int i;
        double result, var;
        result = p.coef[0];         // since x^0 = 1, hence p.coef[0] * x^0 = p.coef[0]
        var = x;
        for(i=1; i<=p.deg; i=i+1) {
                result = result + p.coef[i] * var;
                var = var * x;
        }
        printf("The Evaluated Value is: %lf\n", result);
        return (result);
}
```

Total number of multiplications = 2 + 2 + 2 + . . . (n times) = 2n

Write a function to evaluate a polynomial

Procedure-3: HORNER's RULE

$$a_0 x^0 + a_1 x^1 + a_2 x^2 + a_3 x^3 + a_4 x^4 = a_0 + x (a_1 + x (a_2 + x (a_3 + x (a_4))))$$

```
double Eval3 (Poly p, double x)
{       int i;
        double result;
        result = p.coef[p.deg];
        for(i=p.deg-1; i>=0; i=i-1) {
                result = result * x + p.coef[i];
        }
        printf("The Evaluated Value is: %lf\n", result);
        return (result);
}
```

Total number of multiplications = 1 + 1 + 1 + . . . (n times) = n

Write a function to add two polynomials

Assume: SAME DEGREE of Two Polynomials

```
Poly Add0 (Poly p, Poly q)
{
     Poly r;
     int i;
     r.deg = p.deg; // since p.deg = q.deg
     for(i=0; i<=r.deg; i=i+1) {
          r.coef[i] = p.coef[i] + q.coef[i];
     }
     return (r);
}
```

Write a function to add two polynomials

Handling Two Polynomials with DIFFERENT DEGREES

```
Poly Add1 (Poly p, Poly q)
{
    int i;  Poly r;
    if (p.deg >= q.deg) {
        r.deg = p.deg;
        for(i=0; i<=q.deg; i=i+1) {
            r.coef[i] =
              p.coef[i] + q.coef[i];
        }
        for(i=q.deg+1;i<=r.deg;i=i+1) {
            r.coef[i] = p.coef[i];
        }
    }
```

```
else {
    r.deg = q.deg;
    for(i=0; i<=p.deg; i=i+1) {
        r.coef[i] = p.coef[i] + q.coef[i];
    }
    for(i=p.deg+1;i<=r.deg;i=i+1) {
        r.coef[i] = q.coef[i];
    }
}
return (r);
}
```

Write a function to add two polynomials

Handling Two Polynomials with DIFFERENT DEGREES

Any Other Solution Possible ??

```
Poly Add2 (Poly p, Poly q)
{
      int i;  Poly r;
      if (p.deg >= q.deg) {
            r = p;
            for(i=0; i<=q.deg; i=i+1) {
                        r.coef[i] = r.coef[i] + q.coef[i];
            }
      }
      else {
            r = q;
            for(i=0; i<=p.deg; i=i+1) {
                        r.coef[i] = r.coef[i] + p.coef[i];
            }
      }
      return (r);
}
```