1. Se desea representar un polinomio a través de la entidad como la siguiente. Terminar de implementar el UDT.

```
class Polinomio {
    public:
        // construye un polinomio con los coeficientes y grado dados que
        // representa al polinomio:
        // coefs[0] + coefs[1]*x + coefs[2]*x^2 + ... + coefs[n]*x^n
        Polinomio(const double *coefs, int n);
        Polinomio(const Polinomio &p);
        ~Polinomio();

        Polinomio &operator=(const Polinomio &p);

        Polinomio operator+(const Polinomio &p);
        Polinomio operator*(const Polinomio &p);

        double evaluate(double x); // evalua el polinomio en el valor x

        Polinomio derivate();      // crea y retorna el polinomio derivado

        // crea y retorna el polinomio integrado con constante 'c'
        Polinomio integrate(double c);

    private:
        // puntero a coeficientes
        double *coefs;
        int     grado;
};
```

2. Una empresa que realiza liquidaciones de sueldo en efectivo, requiere un sistema para el cálculo de las cantidades de billetes / monedas de las diferentes denominaciones que hacen falta para pagar una cantidad dada de sueldos. Estas cantidades serán luego solicitadas al banco.
   El sistema planteado, define las siguientes entidades:

```
class Monedero {
    public:
        // construye un monedero con las distintas denominaciones
        // a manejar expresadas en centavos. Por ejemplo si se van a manejar
        // las siguientes denominaciones: $0.50, $1, $2, $10, $50, $100 y $500
        // podría construirse con el vector
        // {50, 100, 200, 2000, 5000, 10000, 50000
        Monedero(const vector<int> &valorMonedas);

        // Agrega un sueldo a ser distribuido en las distintas
        // denominaciones, se debe minimizar la cantidad de
        // billetes/monedas a entregar
        void agregaSueldo(double montoSueldo);

        // Retorna la cantidad de billetes/monedas necesarios de la
        // denominacion de valor 'denominacion'
        int cantidadValor(int denominacion);

    private:
        struct Denominacion {
            int valor;    // valor de la denominación en centavos
            int cantidad; // cantidad de billetes/moneda necesarios
        };
        vector<Denominacion> monedas; // denominaciones a utilizar
```

```
};
```

Se solicita terminar de implementar el UDT. Evalúe la necesidad de implementar un constructor de copia, un operador de asignación y un destructor. En el caso de llegar a la conclusión de que hacen falta, impleméntelos.

3. Se desea representar un polígono con una interface simple para poder conocer algunas propiedades del mismo. La representación e interface deseada es la siguiente. Se solicita completar la implementación del UDT.

```
struct Punto2D {
    double X;
    double Y;
};

struct Poligono {
    public:
        // construye un poligono cerrado con los vertices dados
        Poligono(const vector<Punto2D> &vertices);

        // retorna el perimetro del poligono
        double perimetro();

        // retorna el area del poligono
        double area();

    private:
        vector<Punto2D> vertices;
};
```

Evalúe la necesidad de implementar un constructor de copia, uno de asignación y un destructor. En el caso de llegar a la conclusión de que hacen falta, impleméntelos.