1. Dada la implementación de Stacks de enteros realizada en clase, implementar una función que filtre (elimine) todos los elementos iguales al argumento 'e'. La implementación de la función solicitada debe utilizar únicamente funciones de la interface externa (**`StackPush`**, **`StackPop`**, **`StackIsEmpty`**).

```
void StackFilter(Stack_t *pStk, int e);
```

2. Una forma de analizar la "correctitud sintáctica" de una expresión algebraica en lo que respecta a apertura y cierre de signos de agrupación (paréntesis, corchetes y llaves) es utilizar un stack y:
   a. Recorrer la expresión (string) caracter a caracter.
      i. Cuando se encuentra una apertura de agrupación, se agrega al stack el signo de apertura.
      ii. Cuando se encuentra un cierre de agrupación, el stack no puede estar vacío y el signo que se remueve del mismo se debe corresponder con el signo de cierre que se está procesando.
   b. Cuando se completa el recorrido el stack debe quedar vacío.

   Con esta información, implementar la función:

```
int CheckExpresion(const char *expr);
```

   Probar la función con expresiones como:

   A+Y*{12+z*[sin(34+PI*(2+Y)) + 3] + Q} + (4*Y)

   U otras que tengan errores.

3. Implementar la interface de manipulación de una cola como la vista en la clase dándole a los atributos **`tail`** y **`head`** de la estructura **`Queue_t`**:
   a. Semántica de índices directos sobre el arreglo de elementos.
   b. Semántica de contadores de **`put`** y **`get`** de elementos.

   Probar que las implementaciones funcionan correctamente en los 2 casos.

4. 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:

```
typedef struct {
    int valor; // valor de la denominación en centavos
    int cantidad; // cantidad de billetes/moneda necesarios
} Denominacion;
```

```
typedef struct {
    int numDenominaciones; // cantidad de denominaciones
    Denominacion *pMonedas; // denominaciones a utilizar
} Monedero;
```

y los siguientes servicios, que deben ser implementados:

```
// Crea e inicializa un monedero con capacidad para manejar
// numDenom denominaciones
Monedero *CreaMonedero(int numDenom);

// Destruye un monedero liberando todos los recursos de memoria
void DestruyeMonedero(Monedero *pMonedero);

// Inicializa la entrada indice del monedero para manejar
// la denominación de valor valor (en centavos). Este servicio
// debe ser llamado con valores crecientes de valor.
int SetDenominacion(Monedero *pMonedero, int indice, int valor);

// Agrega un sueldo a ser distribuido en las distintas
// denominaciones, apuntando a minimizar la cantidad de
// billetes/moneda a entregar
int AgregaSueldo(Monedero *pMonedero, double montoSueldo);

// Retorna la cantidad de billetes/monedas necesarios de la
// denominacion de valor valor
int CantidadValor(Monedero *pMonedero, int valor);
```