Se tiene un sistema de gestión de venta de automóviles. Para ello se usan las siguientes estructuras:

```
typedef struct {
   automovil dato;
   struct nodoAuto * siguiente;
} nodoAuto;
```

```
typedef struct {
   char patente[6];
   int valor;
   char marca[10];
} automovil;
```

Realizar las siguientes funciones:

1) nodoAuto * **crearNodo** (automovil auto).
Crea un nodo de tipo nodoAuto *.

2) void **agregarFinal**(**nodoAuto * * lista**, nodoAuto * nuevo).
Agrega un nuevo nodo al final de la lista (utilizando puntero doble)

3) int **insertarCelda**(automovil A [ ], int dim, automovil dato, int validos). Inserta el nuevo automóvil en el arreglo, de manera de conservarlo ordenado por **valor** en forma creciente. **válidos** es la cantidad de datos que tiene el arreglo y retorna validos+1. Al inicio **validos** vale 0.

4) void **agregarMuchos**(**nodoAuto * *** lista).
Esta función pide al usuario el ingreso de los datos de un automovil, crea un nodo y lo agrega al final de la lista (invocando a la función 2).

---

5) int **pasar**(nodoAuto * lista, automovil A[ ], int dim).
Esta función pasa el contenido de la lista al arreglo A, de forma de crear un arreglo ordenado. Para ello debe recorrer la lista e insertar los datos al arreglo usando la función 3.

6) void **mostrarArregloRecursivo**(automovil A[ ], int i, int pos).
Muestra el arreglo en forma recursiva, desde la posición cero hasta la posición pos. El parámetro i representa la posición actual del arreglo a mostrar.

7) Hacer una función **recursiva** que **sume** el valor de los automóviles de la lista que tengan patente “**par**”. Además, deberá pensar una **función** que determine si la patente es **par o no**. (Aclaración: las patentes tienen el formato “**AAA123**”, piense cómo hará para evaluar si el componente numérico es par o impar).

8) Hacer una función **main**() para usar lo anterior.