Nombre: _____

Instrucciones: Escriba sus respuestas a los ejercicios en el orden en que están numerados. Escriba de la forma más clara posible, al finalizar, doble el lado mas largo de la hoja por la mitad junto con sus hojas de respuestas.

1 Parte 1

El objetivo de las preguntas que se pide codificar y responder está relacionado a la programación de funciones para trabajar con matrices ralas.
Una **matríz rala o dispersa** es una matríz donde la mayoría de las entradas son 0. Considere los siguientes datos:

4 5 6

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & -3.6 \\ 9.2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

El primer número (4) es el número de renglones, el segundo (5) el de columnas y el tercero (6) es el número de elementos diferentes de 0.

1. Una forma de mejorar el almacenamiento y eficiencia de métodos que trabajan con estas matrices es almacenar *sólo* los elementos diferentes de 0. Defina un tipo estructura *ELEMENTO* que almacene un elemento de la matríz. La estructura contendrá la posición (renglón y columna) y el valor (número real) de *un* elemento.

2. Defina una estrúctura *MATRIZRALA* que almacene: el número de renglones, número de columnas y número de elementos diferentes de cero, y un arreglo dinámico de tipo ELEMENTO donde se almacenarán los elementos diferentes de cero de la matriz.

3. Escriba una función que recibe de entrada el nombre de un archivo (como cadena de caracteres), y devuelva un apuntador a MATRIZRALA. La función lee una matríz rala como la mostrada arriba, es decir, primero lee el número de renglones, número de columnas y elementos no cero y después la matriz. La función debe de requerir memoria dinámica *sólo* para los elementos diferentes de cero, y debe almacenar solo estos elementos. También requiere memoria dinámica para una estructura MATRIZRALA donde se tendrá acceso a todos los datos de la matríz.

4. Escriba una función para devolver la memoria de la MATRIZRALA del ejercicio anterior, recibe solo el apuntador a la matriz rala y no devuelve nada.

2 Parte 2

1. Una estructura NODO contiene una cadena de caracteres llamada **name** de 128 caracteres, un entero llamado **age** y un apuntador **ptr** a una estructura del tipo NODO. La función **scan(NPTR raiz)**en el código de abajo, lee desde la entrada estándar datos para crear una lista de nodos.

   Escriba la función **creaNodo(·)** mostrada abajo, que requiere memoria e inicializa los campos, con los datos leidos abajo.

```
typedef NODO* NPTR;
NPTR scan(NPTR raiz)
{
  NODO nodo;
   scanf("%s %d", nodo.name,&nodo.age);
   if (raiz==NULL){
        raiz=creaNodo(nodo);
        scan(raiz);
   }
   else if (raiz->age>nodo.age){
        raiz->ptr=creaNodo(nodo);
        scan(raiz->ptr);;
   }
 return raiz;
}
```

2. Considere el siguiente programa que utiliza la función anterior.

```
int main()
{
  NODO *raiz;
   raiz=scan(NULL);
   return 0;
}
```

Dados los siguientes datos desde la entrada estándar:

```
Juan 30
Pepe 29
Luis 28
Pancho -10
Pedro 123
```

¿Cuales son los datos de la lista apuntada por **raiz** después de la función **scan(NULL)** del programa anterior?

3. Escriba una función recursiva **NPTR repara_imprime(NPTR raiz)** que elimine datos en la lista en los que **age** es menor que 0 (por ejemplo, el dato de **Pancho -10** en la pregunta anterior) e imprima el contenido de la lista (sin considerar el dato con age<0).