# PRIMER PARCIAL

## I. Preguntas conceptuales (15 %)

### 1. Explique qué es un registro y de dos ejemplos de la vida real en que se podrían ocupar.

Podemos denominar que un registro es un **tipo** de dato que está a su vez formado por la unión de datos que se denominan campos del registro. Cada campo tiene su propio tipo de dato que es independiente del tipo de los demás campos, por lo que podemos tener un registro que tenga campos de tipo int, float, char, string, etc. sin estar estrictamente limitados a utilizar un tipo específico de datos para todos los campos.

El primer ejemplo de registros en la vida real sería un carné estudiantil, el cual contiene el nombre, apellido, carrera, facultad, año y el ciclo que está cursando.

```
struct carnet{
    string nombre;
    string apellido;
    string carrera;
    string facultad;
    int year;
    int ciclo;
};
```

El segundo ejemplo sería un registro para un sistema de transacciones, este contiene el numero de cuenta, el día, mes y año de la transacción y el monto.

```
struct transacciones{
   int numeroCuenta;
   int dia;
   int mes;
   int year;
   float monto;
};
```

### 3. Si p es un puntero hacia un registro y campo3 es uno de los campos de ese registro, ¿cuál es la forma correcta de acceder a él?

Se tiene 2 maneras para acceder a ese campo del registro a través del puntero, ya que existe la equivalencia de operadores, por lo que se puede acceder de las siguientes formas:

- **p - > campo3**
- **(*p).campo3**

## II. Análisis de código (20 %)

### 1. ¿Cuál función escogió usted?

Función recursiva: Mostrar lista enlazada

```
1  void mostrar(struct nodo *p){
2      if (p == NULL){
3          cout << endl;
4          return;
5      }
6      else {
7          cout << p->dato << " ";
8          mostrar(p->siguiente);
9      }
10 }
```

### 2. ¿Cuántos casos base posee?

Solo tiene un caso base, en este caso es cuando p == NULL el cual se encuentra a partir en la línea de código 2 y termina en la línea 5.

```
2      if (p == NULL){
3          cout << endl;
4          return;
5      }
```

### 3. ¿Cuántos casos recursivos posee?

Posee solamente un caso recursivo, el cual se encuentra en a partir de la línea 6 terminando en la línea 9.

```
6      else {
7          cout << p->dato << " ";
8          mostrar(p->siguiente);
9      }
```

### 4. ¿Qué tipo de recursión utiliza?

Utiliza la recursión lineal final, lineal porque cada llamada recursiva genera, como mucho, otra llamada recursiva, y final porque no queda trabajo por hacer al terminar la llamada recursiva por lo que ese resultado es el que es devuelto por la función.